情 報 保 護 學 會 誌
第 14 券 第 4 號, 2004. 8

안전한 웹 서비스를 위한 웹 어플리케이션 공격 유형 및 대응 방안 분석

이 용 호*, 박 명 수**, 윤 준***, 윤 정 원*

요 약

최근 정보통신 기술의 발달에 따라 인터넷을 통해 다양한 서비스를 제공하는 웹 서비스가 보편화 되고 있으며 이러한 웹 서비스를 가능하게 하는 핵심 요소가 바로 웹 어플리케이션이다. 웹 서비스는 이러한 여러 웹 어플리케이션들이 상호간에 유기적으로 통합되고 연동됨으로써 구성되어진다. 오늘날, 웹 서비스는 정치, 경제, 사회, 문화 등 모든 부분과 밀접하게 관련되어 우리 생활의 모습을 변화시켜가고 있다. 이에 따라 웹 어플리케이션 보안의 중요성이 부각되고 있으며 이에 대한 연구 개발이 필요하다.

본 고에서는 현재 웹 서비스의 해킹 현황을 살펴보고, 웹 어플리케이션 보안의 문제점과 널리 사용되고 있는 공격 유형을 분석한다. 그리고 웹 어플리케이션을 공격으로부터 보호할 수 있는 대응 방안을 살펴본다.

Ⅰ. 서 론

IT 혁명으로 일컬어지고 있는 정보 기술의 발달은 인터넷을 통해 단순한 정보 전달의 수준을 벗어나 전자상거래 및 기업의 업무 처리도 가능하도록 진화해 나가고 있다. 이러한 정보기술 변화를 주도해 나가고 있는 핵심 요소가 웹 어플리케이션이다. 현재 웹 어플리케이션은 전용 소프트웨어를 통해서 이루어지던 모든 개인 및 기업의 업무 형태를 웹이라는 통일된 플랫폼 하에서 시간과 장소에 구애받지 않고 처리할 수 있도록 하고 있다. 이미 마이크로소프트, IBM, SUN 등 세계 유수의 소프트웨어 및 SI 기업들은 웹 기반 기술을 통한 각종 어플리케이션의 통합 및 서비스 제공에 기업 전략의 초점을 맞추고 제품 및 기술 개발에 핵심 역량을 결집시키고 있다. 국내에서도 세계 최고 수준의 초고속정보통신 인프라 보급을 기반으로 웹 기반의 전자상거래, 온라인 쇼핑, 인터넷 포탈, 인트라넷, e-Business 솔루션 개발 등이 이미 보편화 되었고, 전자정부 구축 사업 등의 범국가적인 범위로까지 확대되어 웹 서비스를 통하지 않는 생활을 상상하기 어렵게 되었다. 이와 같이 웹 어플리케이션은 우리에게 많은 편리한 혜택들을 주고 있지만, 그 반면에 지속적으로 보안을 고려하지 않으면 중요한 정보의 노출 및 전송데이터의 위·변조, 업무 장애 등을 통해 큰 피해를 줄 수도 있다.

최근 많은 보안 관련 기업들이 웹 어플리케이션 보안에 관심을 가지고 활발히 활동하고 있다. 웹 어플리케이션 보안에 관심을 가지고 있는 기업들과 전문가들은 OWASP(The Open Web Application Security Project)를 구성하여 가이드라인, 점검리스트 등의 문서 자료와 각종 보안툴을 개발하여 제공하고 있다. 또한 SPI Dynamics, Sanctum, KaVaDo 등의 보안 업체에서는 웹 어플리케이션 전용 취약점 분석 Scanner와 방화벽 시스템을 개발하여 상용화하고 있다. 이러한 노력들은 차세대 정보기술의 기반으로 웹 서비스가 발전하는데 필수적이라 할 수 있다. 또한, 앞으로 웹 기반 기술들이 통합되고, 해킹 기술들이 보다 다양하고 지능적으로 진화함에 따라 이에 대응할 수 있는 웹 어플리케이션 보안에 대한 많은 연구가 필요하다.

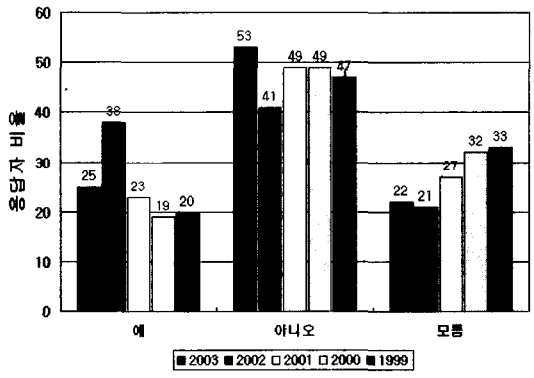
본 논문의 구성은 2장에서는 웹 서비스의 해킹 현황에 대해 언급하고, 3장에서는 웹 어플리케이션 공격 유형을

* 한국전산원 정보화기획단 (leeyh@nca.or.kr, yjw@nca.or.kr)
** 한국전산원 정보화기반구축단 (parkms@nca.or.kr)
*** 한국정보보호인증원 취약성 분석팀(jun@kisa.or.kr)

분석한 후 4장에서 이의 대응 방안 제시하고, 5장에서 결론을 내린다.

II. 웹 서비스 해킹 현황

CSI 와 FBI는 미국 내 기업, 정부기관, 금융기관, 의료기관 및 대학의 보안 업무 종사자 약 500명을 대상으로 사이버 범죄 통계를 조사한 computer crime and security survey를 연차별로 발간하고 있다[1][2]. 2003년 보고서에 따르면 웹 사이트가 침해당한 경험이 있다는 응답자의 비율이 1999년 20%에서 2003년 25%로, 급격한 증가를 보인 2002년의 경우를 제외하면 꾸준히 증가하는 현상을 보이고 있다. 이는 전체 침해 사고 발생 비율이 계속 감소하고 있는 상황을 고려할 때, 최근 침해 사고 대상이 웹 서비스 쪽으로 변화해가고 있는 것으로 보이며 앞으로도 이 추세는 계속될 것으로 판단된다. 웹사이트 침해사고 발생율을 [그림 1]에 나타내었다.

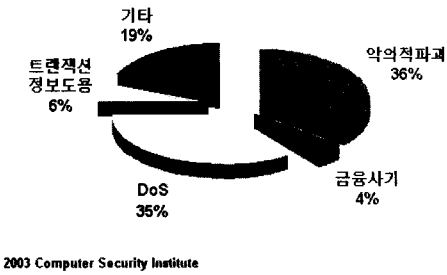


[그림 1] 1년간 웹사이트 침해사고 발생률

웹 공격이 발생하는 진원지를 살펴보면 외부로부터 공격이 발생한다는 응답이 약 50% 정도로 많은 비율을 차지했으나 내부 또는 내·외부 모두에서 발생한다는 응답 비율도 20~30%를 차지해 내부 공격의 비율도 적지 않음을 보여주고 있다. 따라서 웹 공격에 대한 보안을 고려할 때 외부에 의한 공격뿐만 아니라 내부로부터의 공격도 중요하게 고려되어야 할 것으로 판단된다.

[그림 2]에서 나타난 바와 같이 침해사고 유형을 보면 손상 및 변조, DoS가 70% 이상을 차지하고 있으나 금융사기나 트랜잭션 정보 도용과 같이 심각한 결과를 초래할 수 있는 경우도 10%에 이르고 있다.

2004년 보고서에서는 전체 응답자의 7% 가량이 웹 사이트 훼손을 경험했고, 웹 어플리케이션에 대한 공격도



[그림 2] 2003년도 침해사고 유형

10% 정도가 경험한 것으로 나타났다. 이들 공격으로 인한 평가 피해액은 각각 $958,100와 $2,747,000로 전체 평가 피해액 $141,496,560의 2.6%정도이나, 개인 및 기업 정보 유출 등의 평가할 수 없는 무형적 자산까지 고려하면 그 피해 규모는 훨씬 클 것으로 예상된다

따라서 향후 전자정부, 전자상거래, e-Business등 우리 생활의 핵심 IT 인프라 역할을 담당할 웹 서비스의 보안은 반드시 고려해야할 필수 요소이다.

III. 웹 어플리케이션의 공격 유형 분석

1. 웹 어플리케이션 보안의 문제점

웹 어플리케이션 보안의 문제점은 개발 및 관리 단계에서 발생하는 문제점과 본질적인 특징에 따른 문제점으로 나누어 살펴볼 수 있다.

개발 단계에서 발생하는 문제점은 개발자의 보안에 대한 인식과 지식의 부족에 기인한 경우가 많다. 대부분의 개발자들은 어플리케이션의 기능에 대한 지식은 풍부한 반면 보안에 대한 지식은 부족한 경우가 많다. 이에 따라 웹 어플리케이션의 설계 단계에서부터 보안에 대한 고려사항들은 간과하고 지나치는 경우가 많으며, 또한 보안을 고려하고 설계한 경우라 하더라도 개발이 진행되는 과정에서 이에 대한 검증을 생략하는 경우가 발생한다. 웹 어플리케이션 개발 검증 단계에서 사용할 수 있는 자동화된 보안 검증 도구도 부족하다. 이러한 경향은 웹 어플리케이션의 규모가 커짐에 따라 두드러지게 나타난다. 개발 규모가 커지면서 여러 팀으로 나뉘어 개발하거나 외부에 의뢰해서 개발을 하는 경우가 증가하게 되어 개발 시 기능적인 측면을 강조하게 되고 보안적인 측면은 소홀히 하기 쉽다. 개발 과정에서 공개된 sample 코드를 그대로 사용하는 것도 문제가 될 수 있다. 공개된 sample 코드는 일반적으로 보안을 고려하지 않고 만들어진 경우가 많고, 취약점도 이미 노출되어 있을 가능성이 있기 때문에

사용에 많은 주의를 기울여야 한다.

관리 단계에서의 문제점으로는 관리자의 웹 어플리케이션 지식의 부족을 들 수 있다. 일반적으로 보안에 대해 잘 알고 있는 관리자라도 관리하고 있는 시스템의 웹 어플리케이션에 대한 지식은 부족한 경우가 많다. 이 경우, 관리자가 네트웍이나 시스템은 세밀하게 조사하여 적절한 보안 조치를 취하게 되지만 웹 어플리케이션이션에 대한 조사는 생략하기 쉽다. 따라서 웹 어플리케이션의 취약점에 대한 패치 적용도 소홀히 하는 경우가 있다. 관리자의 업무가 반복적이고 업무량이 많아 습관적으로 업무를 처리하게 되는 것도 보안상 문제가 발생하는 원인이 될 수 있다.

웹 어플리케이션 특성상 본질적으로 가지게 되는 문제점들도 있다. 웹은 누구에게나 개방되어 있는 네트웍 포트를 사용하기 때문에 외부 공격에 쉽게 노출된다. 전통적인 방화벽 장비로는 정상적인 웹 트래픽과 해킹을 위한 웹 트래픽을 분류하여 차단할 수 없기 때문에 웹 어플리케이션을 위한 보다 효과적인 보안 수단을 제공하지 못한다. 또한 IDS의 경우도 잘 알려진 웹 어플리케이션에 대한 공격 패턴을 탐지할 수는 있으나 개별적으로 개발된 웹 어플리케이션에 대한 공격 패턴은 탐지할 수 없다. 웹 어플리케이션은 다양한 기능들이 상호 연동되므로 많은 취약점이 존재하여 이에 대한 공격 도구가 많이 존재한다. 이와 같이 해킹을 위해 높은 수준의 지식이 필요하지 않아 초보자들도 쉽게 공격할 수 있다는 점도 주요 문제점으로 지적될 수 있다.

2. 웹 어플리케이션 공격 단계

웹 어플리케이션에 대한 공격은 일반적으로 다음 과정을 거치게 된다. [3]

1단계 : 공격대상 탐색

공격자는 공격 대상 서버에 웹 서비스와 관련하여 어떠한 네트웍 포트[http(80), https(443)]가 열려져 있는지를 검사하고, 검색된 port를 통해 기본 웹 페이지를 내려받는다.

2단계 : 정보 수집

공격자는 웹 서비스를 제공하는 서버의 종류를 파악하고 1단계에서 획득한 기본 웹 페이지의 링크 정보를 분석한다. 이를 통해 공격자는 웹 사이트의 구조와 웹 어플리케이션의 논리 구조를 판단할 수 있게 된다. 또한, 공격자는 수집 가능한 모든 웹 페이지를 분석하여 내부에 숨겨져 있거나 서비스를 제공하지 않는 파일이나 디렉토리의 존재 여부를 파악한다. 정보 수집 단계에서는 수동적인 방법으로 정보를 수집하기 때문에 공격 징후를 남기지 않게 된다.

3단계 : 시험

공격자는 어플리케이션 스크립트나 어플리케이션의 동적인 기능들에 대해 공격 가능 여부를 시험해 보고 개발 과정에서 실수로 관리자 접속 권한을 획득할 수 있도록 허점을 만들지 않았는지를 조사한다.

4단계 : 공격 계획 수립

공격자는 수집된 정보를 확인하여 시험해 본 후에 적절한 공격 방법 및 순서를 결정하게 된다.

5단계 : 웹 어플리케이션 공격

공격자는 수립된 공격 계획에 따라 웹 어플리케이션을 공격한다.

[그림 3]은 웹 어플리케이션의 공격단계를 도식화하여 나타낸 것이다.

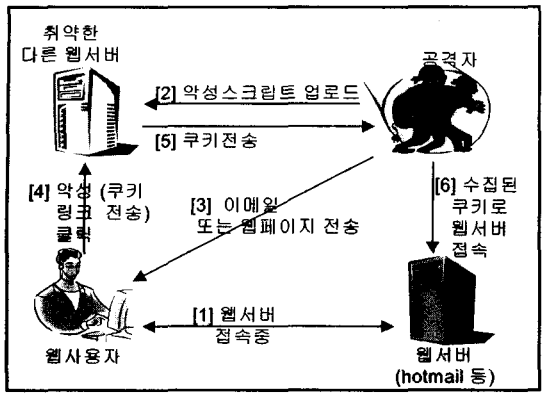
[그림 3] 웹 어플리케이션 공격 단계

3. 웹 어플리케이션 공격 유형

3.1 Cross-site Scripting(XSS)

Cross-site Scripting(이하 XSS) 취약점은 웹어플리케이션이 사용자의 입력으로 받은 악성코드를 필터링하지 않고 그대로 동적으로 생성된 웹페이지에 포함하여 사용자에게 재전송하는 것이다. 자바스크립트처럼 클라이언트측에서 실행되는 언어로 작성된 코드를 사용자 입력으로 주게 되면, 이 코드가 그대로 클라이언트에서 수행되어 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.

예를 들면 [그림 4]와 같다. 웹사용자가 웹서버에 접속중이라고 가정하자 공격자는 XSS 취약점이 취약한 다



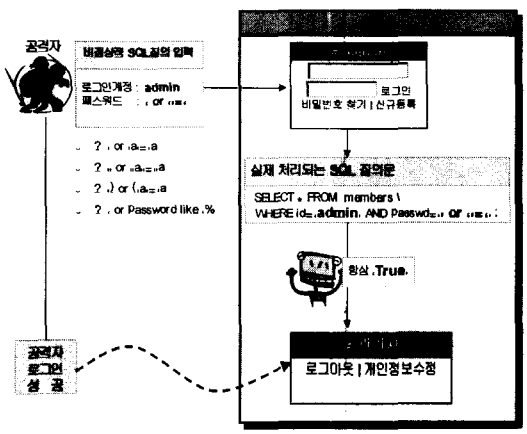
〔그림 4〕 XSS 공격의 예

른 웹서버에 악성스크립트를 업로드한 후 웹사용자에게 이메일 또는 웹페이지를 전송하여 클릭할 수 있도록 유도한다. 웹사용자는 해당 링크를 클릭하고, 취약한 웹서버의 업로드 된 악성스크립트를 통해 해당사용자의 쿠키가 공격자에게 전송된다. 공격자는 수집된 쿠키를 통해 해당 사용자권한으로 웹서버에 로그인하여 제어할 수 있게 된다.

3.2 SQL Injection

최근 웹프로그램은 자료의 효율적인 저장 및 검색을 위해 PHP, JSP, ASP등의 스크립트 언어와 DBMS (DataBase Management System)를 연동하여 사용하는데, SQL Injection은 공격자가 백엔드의 연동된 DBMS를 열람, 수정, 삭제하도록 SQL질의를 수정하여 실행하는 것이다.

SQL Injection 취약점이 있는 웹 어플리케이션은 비정상적인 입력값을 검증하지 않아 사용자 인증을 우회하거나, 데이터베이스에 저장된 데이터를 열람해 볼 수 있다.



〔그림 5〕 SQL Injection 공격의 예

SQL Injection 인증우회의 예를 들면 〔그림 5〕와 같다. 공격자는 비정상적인 SQL 질의를 웹서버의 로그인 입력부분에 유추된 로그인계정인 'admin'과 패스워드로 ' 'or ''=' ' 값을 입력한다. 실제 처리는 『SELECT * FROM members WHERE id='admin' AND passwd='' or ''='' 』와 같은 SQL 질의문으로 처리된다. or 이하의 SQL질의문이 항상 옳은 값이 되므로 공격자는 패스워드를 모르는 상태에서도 관리자로 로그인이 가능하게 된다.

3.3 버퍼오버플로우 취약점

버퍼오버플로우 취약점은 일반적인 응용프로그램 보안 취약점의 하나로써 대표적인 웹어플리케이션의 취약점에 포함된다. 프로그램에서 버퍼의 한계를 점검하지 않고 작성된 코드부분을 이용하여 악의적인 공격자 코드로 리턴 어드레스 등을 덮어쓰도록 해서 프로그램의 정상적인 동작을 변경하거나 프로그램이 다운되도록 하는 공격 방법이다.

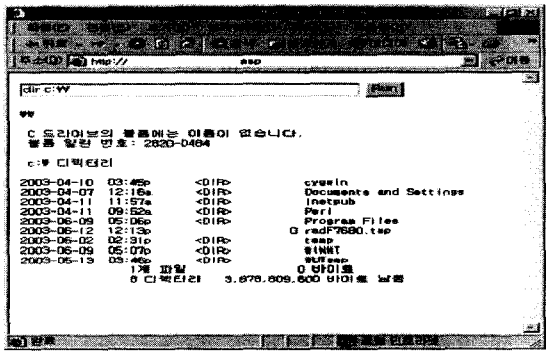
〔표 1〕는 버퍼오버플로우의 개념적인 예를 보여주기 위한 코드이다.

256개 크기의 배열 src를 생성하고, 배열의 반은 'A'로 나머지 반은 'B'로 초기화시킨다(13라인~19라인). 배열 src를 인자로 함수를 호출한다(20라인). 함수 내에서 128개 크기의 배열 dst를 만든 후 배열 dst에 배열 src를 복사한다(6라인~8라인). 일반적으로 프로그

〔표 1〕 버퍼오버플로우의 예

```
1:  #include <stdio.h>
2:  #define MAX 256
3:
4:  char *func(char *s, char *d)
5:  {
6:    char *dst[MAX/2];     int i;
7:    for( i=0; i<MAX; i++)
8:      d[i]=s[i];
9:    return d;
10: }
11: int main(int argc, char *argv[])
12: {
13:   char src[MAX], ;
14:   int i;
15:   for(i = 0; i<MAX/2; i++)
16:     src[i] = 'A';
17:   for( ; i<MAX-1 ; i++)
18:     src[i] = 'B';
19:   src[i] ='\0';
20:   printf("호출값 =%s\n", func(src));
21: }
```

램은 다음과 같은 4가지 영역으로 구성되어 있다. 프로그램 코드(Text), 정적변수 저장되는 부분(Data), 힙(Heap), 스택(Stack)이다. 함수를 호출하게 되면, 프로그램 실행파일의 상위 어드레스 부분에 있는 Stack 영역에 함수내에서 동적으로 할당되는 데이터, 함수내의 변수, 함수의 리턴 어드레스 등이 할당된다. 함수 내에서 복사시 변수영역인 dst의 할당영역을 넘어서 함수의 return Address 영역까지 복사가 되면, 메인함수의 'B'로 초기화된 부분이 return Address를 덮어 쓰게 된다. 이 부분을 악용하여 공격자의 의도에 따른 프로그램을 수행할 여지가 있다.



〔그림 6〕 파일업로드 취약점 공격의 예

3.4 파일 업로드 취약점

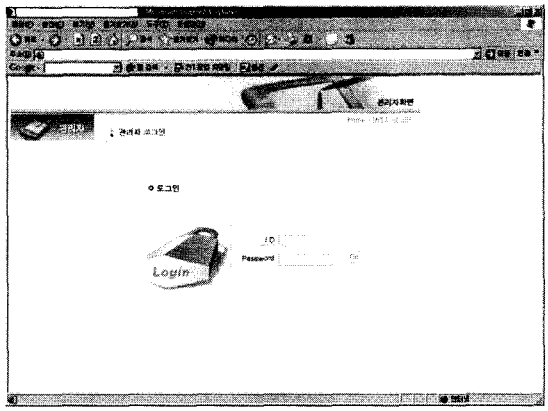
파일 업로드 취약점은 공격자가 게시판의 파일 첨부를 이용해서 웹서버의 사용자 쉘을 획득할 수 있는 취약점이다. 〔그림 6〕은 그 예를 보여준다.

공격자는 악의적인 목적코드가 포함된 파일 (command.asp등)을 게시판이나 자료실에 업로드하여 생성하고 그 파일을 웹서버에게 요청하면, 웹서버는 파일에 포함된 악성코드를 실행한 후에 그 결과를 웹페이지에 담아 공격자에게 보여주게 된다. 만약에 웹서비스가 최상위관리자(또는 루트) 권한으로 수행되면 이 취약점을 공격자도 루트 권한을 갖게 되어 시스템 전체를 사용불가능 상태로 만들어 버릴 수 있다.

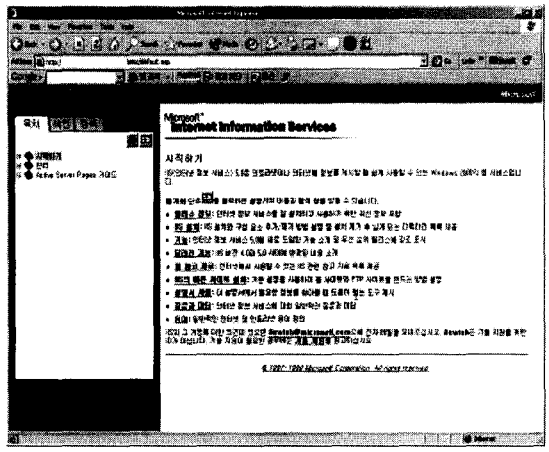
3.5 기타 취약점

기타 웹 서비스 중인 홈페이지에는 다음과 같은 다양한 취약점이 존재할 수 있다.

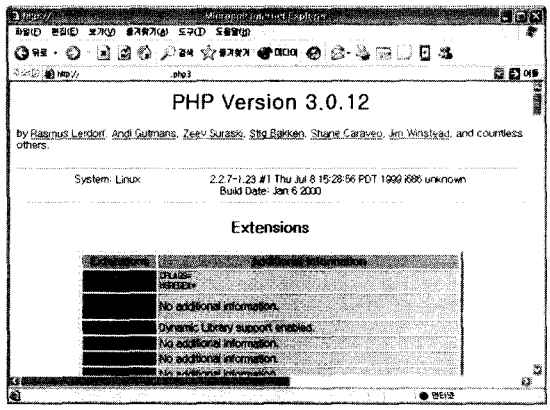
〔그림 7〕과 같이 웹서버의 관리자 페이지를 외부에 노출된 채로 방치한 경우, 공격자는 웹을 통한 전수공격(Brute-Force Attack) 도구를 이용해 손쉽게 웹서버의 관리자 권한을 획득할 수 있다.



〔그림 7〕 관리자 페이지 노출의 예



〔그림 8〕 기본 설치 파일 노출의 예(IIS)

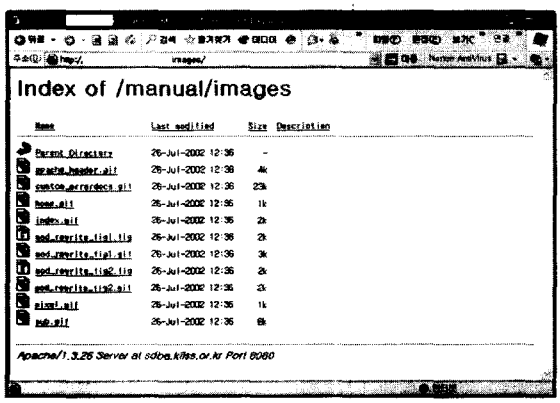


〔그림 9〕 불필요한 시스템정보 노출의 예(PHP)

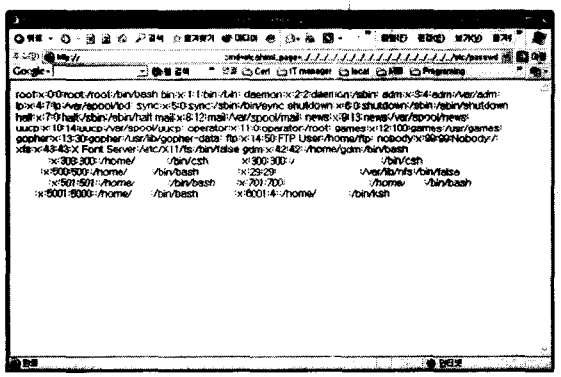
〔그림 8〕과 〔그림 9〕와 같이 웹서버 설치 시에 기본적으로 설치되는 파일이 노출되거나, 서버에서 old, bak

또는 테스트파일과 같은 불필요한 파일이나 현재 서비스 중인 시스템 정보가 노출되어 질 경우 공격자에게 웹서버의 정보수집을 좀 더 용이하게 하여 준다.

〔그림 10〕은 서버의 환경설정 적절하지 않아 파일 정보가 노출되는 취약점을 보여 주고 있다.



〔그림 10〕 부적절한 환경설정으로 노출되는 예(Apache)



〔그림 11〕 지정된 경로 외의 파일 노출(/etc/passwd)

〔그림 11〕와 같이 지정된 경로외의 파일을 다운로드할 수 있거나 패치가 제대로 되지 않은 공개게시판을 이용하는 등 웹서버에는 다양한 보안취약점이 있다.

Ⅳ. 웹 어플리케이션의 공격유형에 따른 안전한 보안대응 방안

1. 웹 어플리케이션 개발 시의 고려할 보안원칙

앞서 기술된 XSS, SQL Injection, 버퍼 오버플로우, 파일 업로드 취약점 및 기타 보안취약점은 웹사이트 개발 시에 보안사항이 고려되지 않아 발생한 문제점들이다.

OWASP에서는 이런 문제점을 사전에 예방하고자 웹 어플리케이션 개발 시에 다음과 같은 보안원칙을 제시하고 있다. 〔6〕 이런 보안원칙이 준수된다면, 보안취약점들은 상당부분 개선되어 질 것이다.

o 사용자의 입력 값과 출력 값을 검증

시스템의 사용자 입력 값과 출력 값에는 제한된 범위 내의 문자들만 허용하고 다른 데이터는 차단시켜야 한다. 예를 들어 입력 값이 주민등록번호인 경우에 13자리 숫자가 아닌 데이터는 비정상적인 것으로 처리해야 한다. 특정 문제를 방지하기 위해 특정 문자열만 필터링하는 것은 잘못된 방법이다. 방화벽이 특정 패킷을 규칙에 의해 차단하고 모든 트래픽을 허용하는 경우를 생각해 보면 적절하지 않음을 알 수 있다.

o 오류 발생시 안전하게 처리 수행

웹 어플리케이션은 오류 발생시에도 안전하게 처리되어야 한다. 웹 어플리케이션이 오류를 발생한 경우 다음에 수행될 함수는 실행되지 않도록 해야 한다. 버퍼오버플로우 공격과 같이 메모리영역의 한계를 초과 한 경우에도 안전하게 처리된다면 공격자로부터 안전할 것이다.

o 단순하게 유지

웹 개발자나 관리자는 보안요구사항에 맞게 보안제어를 복잡하게 추구하려 하나, 실제 사용자는 보안시스템이 너무 복잡하면, 사용하지 않거나 무시해 버릴 수도 있다. 예를 들어 사용자에게 20자리 패스워드 입력을 강요하는 것 등이다. 최상의 보안은 최고로 간결하고 다루기 쉬운 보안시스템을 유지하는 것이다.

o 신뢰할 수 있는 컴포넌트의 활용

일반적으로 시스템 개발자들은 다른 개발자들과 유사한 문제에 직면한다. 당면한 문제에 대해 명확한 해결책을 마련하기 위해, 대부분의 경우 반복적인 작업을 통해 자신만의 라이브러리나 컴포넌트를 활용한다. 그러나, 신뢰할 수 있는 타인의 라이브러리나 컴포넌트를 재사용하는 것도 자원 활용 측면에서 생각할 때 바람직하다.

o 여러 가지 상황에 대응할 수 있는 방어

보안을 위해 한 가지 컴포넌트에 의지하는 것은 바람직하지 않다. 사전에 잘 계획대고 준비된 컴포넌트라 하더라도 다양한 공격 방법을 모두 방어하기는 어렵다. 또한 컴포넌트가 공격에 대응할 때 예측하지 못한 오류가 발생할 수도 있다. 따라서 여러 컴포넌트를 단계적으로

배치하여 한 컴포넌트가 보안 대응에 실패하면 다음 컴포넌트가 이에 대응할 수 있도록 철저히 준비해야 한다.

o 보안은 가장 취약한 부분만큼 안전

보안은 가장 취약한 부분이 안전한 수준만큼 안전하다. 일반적으로 웹사이트에서 128bit SSL을 이용해서 100% 안전하다고 하는 광고를 흔히 볼 수 있다. 그러나, 웹서버와 사용자의 브라우저 간에 전송되는 데이터가 적절한 보안전송방법을 가지고 있는 것이 전체 보안수준을 좌우하는 것이 아니라 사용자의 패스워드가 노출되거나, PC에 보안패치가 제대로 되지 않는 등의 사소한 취약한 점이 전체 보안수준을 결정한다는 것이다. 예를 들면, 실생활에서 정문을 잠그고 뒷문은 열어놓은 채 외출하는 것이 올바르지 않듯이, 자신의 웹서버에 취약한 점을 깊이 고려하여 개발한 시스템만이 전체 시스템의 보안을 좀 더 보장하게 된다. 일반적으로 공격자는 게으르고 가장 쉽게 찾은 취약점을 이용하여 공격하려고 시도하기 때문이다.

o 은폐시키는 보안은 적용되지 않는다.

공격자로부터 정보를 단순히 은폐하는 행위는 시간을 낭비할 뿐이다. 이미 알려진 소프트웨어의 취약점들을 수년 동안 숨겨져 왔다. 드문 경우이기는 하나 이미 파악된 자사 어플리케이션의 취약점을 숨기는 경우도 있다. 그러나, 정보를 은폐하는 것과 보호하는 것은 다른 것이다. 정보를 은폐하는 전략은 오랫동안 지속되지도 않고, 단기간의 도움을 주지도 못한다.

o 최소권한의 원칙

시스템은 작업 수행에 필요한 최소 권한을 가지도록 디자인 되어야 한다. 이것이 "need to know" 접근방식이다. 사용자에게 절대 필요이상의 권한을 할당해서는 안된다. 가정부에게 저녁식사를 준비하게 하기 위해 은행통장 전체를 맡기는 것이 바람직하지 않은 것처럼 적절하게 권한을 부여하는 것이 필요하다.

o 직무분리

직무분리를 통해 사용자, 프로세스, 데이터에 대한 권한을 구분하는 것은 그들이 발생할 수 있는 문제의 범위를 제한하는데 도움을 줄 수 있다. 정보보안에서 광범위하게 적용되고 있는 중요한 개념이다. 풀장관리인에게는 집 열쇠를 맡기고 외출하는 일은 현명하지 않은 처사이다. 관리인에게 풀장에 대한 권한만 제공하여 문제를 발생시킬 수 있는 문제의 범위를 제한시킬 수 있는 것이 옳은 방안이다.

2. 웹 어플리케이션 공격 유형별 보안대응방안

2.1 Cross-site Scripting(XSS)

XSS 취약점을 제거하기 위해서는 수행해야 할 대응방안은 다음과 같다. [7]

<웹서버 개발자 및 관리자>

- 사용자 입력으로 사용 가능한 문자들을 정해놓고, 그 문자들을 제외한 나머지 모든 문자들을 필터링 한다.
- 게시판에서 HTML 포맷의 입력 기능을 사용할 수 없도록 설정한다.
- 개인정보변경 등 중요한 메뉴는 재인증을 요구하도록 한다.
- 웹 취약점 스캐너와 수작업에 의한 취약점 점검방법을 이용해서 웹 서비스의 취약점을 주기적으로 점검한다.

<웹 사용자>

- 의심이 가는 메일이나 게시판의 글을 절대 열지 않도록 한다.
- 웹사이트 방문 시 메일이나 웹문서에 포함된 링크를 클릭하지 말고, 직접 브라우저 주소창에 URL을 입력한다.
- 브라우저에 최신 보안패치를 적용하도록 한다.

웹 서버 개발자나 관리자 입장에서는 사용자 ID, 암호 입력칸, 검색어 입력 칸과 같은 모든 사용자 입력부분을 필터링하도록 해야 한다. 필터링해야 하는 대상은 GET 질의 문자열, POST 데이터, 쿠키, URL 그리고 일반적으로 브라우저와 웹서버가 주고받는 모든 데이터를 포함한다. 클라이언트가 입력한 데이터에서 특수문자를 HTML문자로 바꾸도록 한다.([표-2])

웹사용자의 입장에서 모든 클라이언트에서 수행되는 스크립트 언어가 브라우저나 이메일을 읽을 때 수행되지 않도록 해야 하나, 현실적으로 이 방안은 불가능하다. 그

[표 2] HTML 문자코드표

변경전	<	>	(	)	#	&
변경후	<	>	&#40	&#41	&#35	&#38
변경전	'	"	/	\	;	
변경후	&#39	&#34	&#47	&#92	&#59	
변경전	Line Feed			Carriage Return		
변경후	&#10			&#13		

래서, 웹사용자는 익명의 사람에게 수신된 이메일을 확인하거나, 의심스러운 웹페이지를 방문시에 주의하도록 해야 한다. 사용자의 웹브라우저에 알려진 보안취약점에 대한 최신 보안패치를 정기적인 적용한다.

2.2 SQL Injection

SQL Injection 취약점을 제거하기 위해서 수행해야 할 대응방안은 다음과 같다. [7]

- 사용자 입력이 SQL injection을 발생시키지 않도록 사용자 입력을 필터링 한다.
- SQL 서버의 에러 메시지를 사용자에게 보여주지 않도록 설정한다.
- 웹 어플리케이션이 사용하는 데이터베이스 유저의 권한을 제한시킨다.
- 데이터베이스 서버에 대한 보안 설정을 수행한다.

XSS 취약점과 같이 사용자의 입력값을 SQL 질의문으로 사용되지 않도록 최대한 필터링 하는 방법이 최선의 해결책이다.

2.3 버퍼오버플로우 취약점

버퍼오버플로우 취약점은 제거하기 위해서는 수행해야 할 대응방안은 다음과 같다. [7]

- 웹서버에 대한 최신 취약점 정보에 항상 주의를 기울이고 발표된 패치를 적용한다.
- 직접 개발한 어플리케이션에 대해서는 모든 외부 입력에 대해 적절한 한계 체크를 수행하고, 버퍼오버플로우 취약점이 존재하는 함수의 사용을 금한다.
- 웹서버와 어플리케이션이 제한된 권한을 가진 계정으로 실행되도록 한다.

버퍼오버플로우 취약점은 주기적인 보안패치를 통해 스택에서 실행이 불가능하도록 하고, 웹 어플리케이션 개발 시에 바운드 체크 기능이 없는 함수를 사용을 금한다. C 라이브러리의 *get()*, *strcpy()*, *strcmp()*와 같은 함수는 취약점이 존재하므로 strncpy(), strncmp()와 같은 바운드 체크기능이 있는 함수나 Perl, Python, Java와 같은 자동화된 바운드 체크 기능을 제공하는 언어를 사용하여 개발한다.

2.4 파일 업로드 취약점

파일업로드 취약점을 제거하기 위해서는 수행해야 할 대응방안은 다음과 같다. [7]

- 게시판의 파일첨부 기능을 제거하거나, 문서파일 확장자를 갖는 파일만 업로드가 가능하도록 수정한다.
- 파일 업로드 디렉토리에 대해서는 스크립트 실행이 불가능하도록 설정하고, 가능하다면 업로드된 파일에 대한 실행이 불가능하도록 설정한다.
- 게시판 관련 최신 취약점 정보에 주의를 기울이도록 한다.
- 게시판 외에 외부로부터 파일이 업로드 될 수 있는 모든 곳에 대해 위와 같은 보안조치를 수행한다.
- 파일 업로드 방법외에 사용자가 스크립트 코드를 업로드 시켜서 실행시킬 수 있는 곳이 있는지 점검해본다.

2.5 기타 취약점

관리자 페이지 노출 취약점인 경우 일반적으로 관리자 페이지 경로를 유추하기 어려운 이름으로 변경하고, IP 레벨의 접근 권한을 설정한다. 그리고, 관리자 인증을 위한 세션관리를 통해 해당 페이지를 직접 입력하여 로그인하는 것을 방지한다.

기본 설치 파일과 같은 불필요한 시스템정보 노출되는 취약점은 제거하기 위해서 일반적으로 시스템에서 사용되는 파일 목록이 보이지 않도록 하며, 시스템 설치시 기본적으로 설치되는 운영 시에 불필요한 파일(old, bak 파일 등)은 사전에 삭제 또는 IP레벨의 접근권한을 제어함으로써 사전예방을 할 수 있다.

V. 결 론

본 고에서는 웹 어플리케이션의 전반적인 공격 유형과 OWASP 보안지침 및 공격 유형별 대응방안을 살펴보았다.

웹 어플리케이션의 보안을 위협하는 공격 기법은 점점 더 다양해지고 지능적으로 변화하고 있다. 따라서 이에 대비하기 위해 더 많은 보안 기술들이 연구되고 개발될 필요가 있다. 특히 웹 어플리케이션 보안 기술 개발이 활성화된 외국과는 달리 국내 기술은 아직 초기 단계로, 이에 대한 적극적인 관심 및 제품 개발이 시급하다.

본 고에서 언급된 공격 유형 및 대응 방안들은 향후 안전한 웹 서비스 시스템을 개발하고 구축하는 데뿐만 아니라 웹 어플리케이션 보안 기술 개발에도 참고가 될 수 있을 것이다.

참 고 문 헌

[1] "2003 CSI/FBI Computer Crime and Security Survey", CSI, 2003

[2] "2004 CSI/FBI Computer Crime and Security Survey", CSI, 2004

[3] Caleb Sima, "Security at the Next Level" SPI Dynamics, 2004

[4] David Endler, "The Evolution of Cross-Site Scripting Attacks", iDEFENSE Inc, 2002

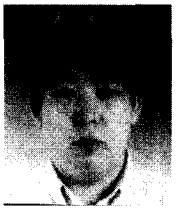
[5] Alphe One, "Phrack49-16 : Smashing The Stack For Fun And Profit", www.phrack.org , 1996

[6] Mark Curphey 외 16명, "A Guide to Building Secure Web Applications:The Open Web Application Security Project", 2002

[7] 윤 준, "최신 웹 해킹기법에 대한 분석과 대응방법", 한국정보보호진흥원, 2004

[8] "홈페이지 주요취약점 및 보호대책", 한국정보보호진흥원, 2004

〈著者紹介〉



이 용 호(Yongho Lee)

1999년 : 연세대학교 전자공학과 석사

1997년~1998년 : 연세대학교 정보통신 연구소 연구원

1999년~2004년 : 두루넷 기술연구소 연구원

2004년~현재 : 한국전산원 국제협력팀 전임연구원

〈관심분야〉 정보보호, 차세대 인터넷, BcN

박 명 수(Myungsu Park)

1999년 : 충남대학교 컴퓨터과학과 학사

2001년 : 충남대학교 컴퓨터과학과 석사

2000년~현재 : 한국전산원 인터넷기반·인증팀 전임연구원

〈관심분야〉 정보보호 및 네트워크 보안, IT아웃소싱 운영 및 ROI

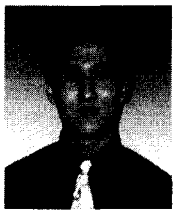
윤 준(Jun Yoon)

1997년 : 충남대학교 컴퓨터과학과 학사

1999년 : 충남대학교 컴퓨터과학과 석사

2000년~현재 : 한국정보보호진흥원 취약성분석팀 연구원

〈관심분야〉 웹 보안, 무선랜 보안, 정보보호 컨설팅



윤 정 원(Jongwon Yoon)

1992년 : 캘리포니아주립대 컴퓨터공학과 학사

1994년 : 캘리포니아주립대 컴퓨터공학과 석사

1992년~1993년 : 캘리포니아주립대 의용공학 연구소 연구원

1999년~2000년 : UN 산하 국제Y2K협력센터 통신부분 조정관

2001년 : 미국 국립표준기술원 초빙연구원

1995년~현재 : 한국전산원 국제협력팀장

〈관심분야〉 PKI, Risk Management, BCP