멀티테넌시 기반 웹 사이트의 OWASP TOP 10 보안취약성 검증 방법

이도현* · 이종욱** · 김점구***

요 약

요즘 웹 애플리케이션의 보안 취약점을 이용한 해킹과 수 많은 사이트에서 개인정보의 노출로 인한 웹 사이트의 보안 문제가 날로 증가하고 있다. 그리고 이로 인한 피해가 날로 높아지고 있어 그에 대한 대책으로 안전한 웹 사이트 제작방법이 절실히 요구되고 있는 상황이다. 이에 본 논문은 웹 사이트의 제작 시에 오픈소스 웹 애플리케이션 보안 프로젝트를 고려한 OWASP TOP 10 취약점 확인방법을 제안하였고, 제안 방법을 적용하여 보안취약점을 검증하는 방법 및 취약점 개선 후 성능에 대해 분석하였다.

Verification Methods of OWASP TOP 10 Security Vulnerability under Multi-Tenancy Web Site's Environments

Do Hyeon Lee* · Jong Wook Lee** · Jeom Goo Kim***

ABSTRACT

Nowadays hacked using a security vulnerability in a web application, and the number of security issues on the web site at many sites due to the exposure of personal information is increasing day by day. In this paper, considering the open-source Web Application Security Project at the time of production of the website. Proposed the OWASP TOP 10 vulnerability verification method, by applying the proposed method and then analyzed for improved method and vulnerability to verify the performance of security vulnerability.

Keyword OWSP, Multi-Tenancy, Website, Vulnerability, Security

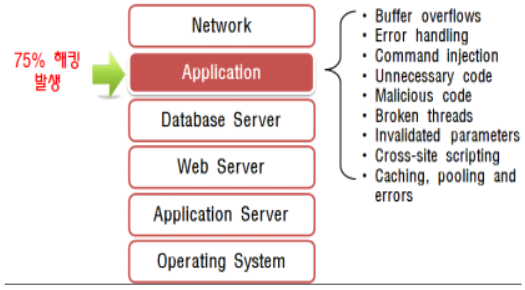
접수일(2016년 05월 31일), 수정일(1차: 2016년 06월 14일), 게재확정일(2016년 06월 20일)

* (주)신성씨엔티
** 한국폴리텍대학교/모바일정보통신과
*** 남서울대학교/컴퓨터학과

1. 서 론

최근 다수의 웹 사이트를 개발, 운영하는 공공기관, 교육기관 및 일반기업은 개별 대상에 맞는 웹 사이트를 별도 개발하던 분산된 개발/관리의 방법에서 변화하여 단일 엔진을 통해 다수의 웹 사이트를 생성하고 관리하는 SaaS (Software as a Service)나 애플리케이션 임대 서비스(ASP)의 방식들로 변화하고 있다. 여기에 클라우드(Cloud) 서비스의 기술진보는 웹 사이트의 개발 및 운영에 통합된 환경 하에서의 개발/관리/운영의 방향으로 변화하고 있다. 특히, 단일 소스를 이용하는 시스템 운영체계인 클라우드 서비스 환경에서의 보안 침해 공유 사이트가 확산되고 있다. 또한 홈페이지를 운영하다 보면 많은 보안 문제들과 사고가 생긴다.

최근의 가트너사 발표에 의하면 (그림 1)과 같이 사이버 공격의 약 75%가 응용 프로그램(즉, SW)의 취약점을 악용한 것이라고 한다[1][2].



(그림 1) 사이버 공격의 분포

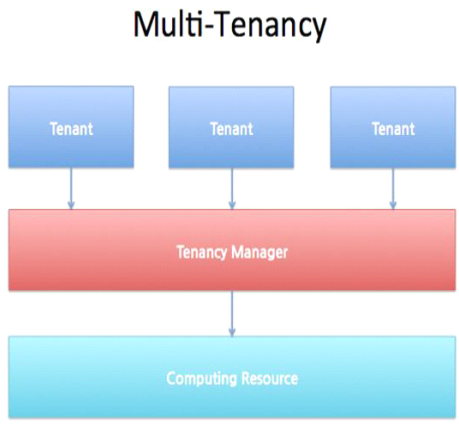
본 논문에서는 홈페이지 제작에서 부터 발생할 수 있는 이러한 웹 애플리케이션 보안취약점과 보안취약점에 대한 개선방법에 대해서 연구하고 보안 취약점 개선 후에 보안 취약점에 대한 검증하는 방법 및 취약점 개선 후의 성능에 대해서 분석을 하고자 한다.

2. 관련연구

2.1 멀티테넌시

멀티테넌시(Multi-Tenancy)는 단일 인스턴스가 다수의 사용자와 다수의 그룹에게 서비스를 제공하기 위한 소프트웨어 아키텍쳐이다[11].

멀티테넌시는 클라우드 컴퓨팅 환경에 적합한 소프트웨어 아키텍쳐로 많은 서비스 애플리케이션에 차용되어 사용되고 있다. 멀티테넌시에서는 각각의 서비스 애플리케이션에서 사용하는 데이터베이스와 같은 컴퓨팅 리소스를 가상 파티션(virtual partition)으로 분리하여 사용자와 그룹에게 제공한다. 이러한 가상 파티션은 해당 서비스 애플리케이션의 사용자에게는 실제의 데이터와 동일하게 제공되기 때문에 애플리케이션 차원에서는 가상 데이터와 실제 데이터와의 차이는 전혀 없이 사용할 수 있다. 테넌시 관리자(tenancy manager)는 이러한 서비스 애플리케이션에서 데이터에 대한 요청이 있을 때 해당 요청의 적법성을 확인하여 허용된 데이터에 액세스 할 수 있도록 관리하는 역할을 한다. (그림 2) 는 멀티테넌시의 일반적인 구조를 보여준다.



(그림 2) 멀티테넌시 일반적 구조

단일 테넌시(single-tenancy)에서는 각 서비스 애플리케이션마다 데이터베이스와 같은 데이터를 독립적으로 가지고 있는 구조로 이루어져 있지만 멀티테넌시에서는 주요 데이터에 대해서 각각의 서비스 애플리케이션들이 공유하여 사용한다. 보안상의 이유나 성능상의 이유로 생각해 볼 때 멀티테넌시보다는 단일 테넌시가 훨씬 우수함에도 불구하고 멀티테넌시가 클라우드 컴퓨팅에서 많이 채택되는 결정적인 이유는 투자수익율 때문이다. 싱글테넌시에서는 각각의 데이

터를 위한 하드웨어가 필요하지만 멀티테넌시에서는 그렇지 않기 때문에 보다 적은 비용으로 서비스운영이 가능해진다는 장점이 있다. 아울러 테넌시 관리자를 이용해 각 서비스 애플리케이션을 제어하고 관리할 수 있다는 점도 장점 중의 하나이다[10].

멀티테넌시의 장점은 멀티테넌시를 위한 애플리케이션의 설정 기능 이외 서버 자원의 공유 측면에서 상당한 비용을 줄일 수 있다는 것이다. 기존의 단일 테넌시 애플리케이션 개발의 한계점은 테넌시들이 가상 서버를 통해 서비스를 제공받기 때문에 테넌시별로 분산된 리소스와 독립된 서버 환경을 가지고 있기 때문에 테넌시의 수가 증가할수록 유지보수 비용이 증가한다는 점이다. 하지만 SaaS에서와 같이 공유 호스팅 환경에서 리소스를 공유하고 하나의 애플리케이션 인스턴스를 통해 멀티테넌시에게 서비스를 제공하면 테넌시의 수가 증가할수록 유지보수 비용은 상대적으로 낮아질 수 있다.

2.2 OWASP

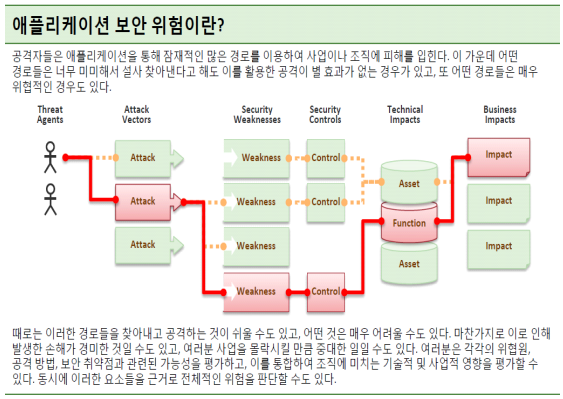
홈페이지 보안 취약점 및 개선방법에서는 OWASP (The Open Web Application Security Project)에서 설명하는 보안 취약점 및 그에 대한 개선방법을 다루고자 한다.

OWASP는 오픈소스 웹 애플리케이션 보안 프로젝트이다. 주로 웹에 관한 정보노출, 악성 파일 및 스크립트, 보안 취약점 등을 연구하며, 10대 웹 애플리케이션의 취약점 (OWASP TOP 10)을 발표했다. OWASP TOP 10은 웹 애플리케이션 취약점 중에서 빈도가 많이 발생하고, 보안상 영향을 크게 줄 수 있는 것들 10가지를 선정하여 3년 주기로 기준이 발표되었고, 문서가 공개되었다[12].

본 논문에서는 OWASP에서 소개한 것 중에서 2013년에 발표된 OWASP TOP 10 2013을 기준으로 소개한다.

(1) OWASP TOP 10

OWASP 에서는 해커들은 애플리케이션을 통해 보안 취약점을 이용해서 웹 사이트에 피해를 입혔다. 이 가운데 어떤 보안취약점들은 너무 미미해서 찾아도 이를 활용한 공격이 별 효과가 없는 경우가 있고 어떤 경우는 매우 위협적인 경우도 있다. 이러한 보안 취약점을 평가하고 동시에 이러한 요소들을 근거로 전체적인 위험을 판단할 수도 있다. 이런 보안 취약점들 중에서 보안상 영향을 크게 줄 수 있는 10가지를 선정한 것들은 (그림 3)과 같다.



(그림 3) 애플리케이션 보안위험이란

3. OWASP TOP 10 보안 취약성 확인 방법

3.1 A1 보안 취약성 확인 방법

인젝션은 SQL, 운영체제, LDAP 인젝션 취약점은 신뢰할 수 없는 데이터가 명령어나 질의문의 일부분으로서 인터프리터로 보내질 때 발생한다. 공격자의 악의적인 데이터는 예상하지 못하는 명령을 실행하거나 적절한 권한 없이 데이터에 접근하도록 인터프리터를 속일 수 있다.

확인방법은 인젝션에 취약한지 알아내는 가장 좋은 방법으로 SQL 호출 시 모든 준비된 구문과 저장된 프로시저에서 바인드(bind) 변수들을 사용하고 동적 질의를 피해야 한다. 안전하게 사용하는지 알아보기 위한 빠르고 정확한 방법은 코드를 검사하는 것이다. 자동화된 동적 스캐닝을 이용해서 애플리케이션 실행하면 몇몇 악용되는 인젝션 결함이 존재하는지 알 수 있다.

3.2 A2 보안 취약성 확인 방법

인증 및 세션 관리 취약점은 인증과 세션 관리와

관련된 애플리케이션 기능은 정확하게 구현되어 있지 않아서 공격자가 패스워드, 키 또는 세션 토큰을 해킹하거나 다른 구현 취약점을 공격하여 다른 사용자 ID로 가장할 수 있다.

확인방법은 사용자 인증 정보 및 세션 ID와 같은 세션 관리 자산은 적절히 보호되고 있는가? 와 같이 다음의 경우에 대해서 검토를 할 필요가 있다.

- 사용자 인증 정보가 저장될 때 해시 또는 암호화를 사용하여 보호되지 않는다. A6 참고.
- 인증 정보가 취약한 계정 관리 기능(예를 들어 계정 생성, 패스워드 변경, 패스워드 복구, 취약한 세션 ID)을 통해 추측되거나 덮어쓰기가 가능하다.)
- 세션 ID가 URL에 노출된다.(예를 들어, URL 다시 쓰기)
- 세션 ID가 세션 고정 공격에 취약하다.
- 세션 ID가 타임아웃 되지 않거나, 사용자 세션 또는 인증 토큰, 특히 싱글 사인온(SSO) 토큰이 로그아웃 된 동안 적절히 무효화 되지 않는다.
- 세션 ID가 성공적인 로그인 이후 교체되지 않는다.
- 패스워드, 세션 ID 및 기타 증명이 암호화되지 않은 연결을 통해 전송된다.

3.3 A3 보안 취약성 확인 방법

크로스 사이트 스크립팅 (XSS)는 애플리케이션이 신뢰할 수 없는 데이터를 가져와 적절한 검증이나 제한 없이 웹 브라우저로 보낼 때 발생한다. 공격자가 피해자의 브라우저에 스크립트를 실행하여 사용자 세션 탈취, 웹 사이트 변조, 악의적인 사이트로 이동할 수 있다.

확인방법은 출력 페이지에서 해당 입력 값을 포함하기 전에 모든 사용자가 제공한 입력이 제대로 이스케이프 되었는지 확인하지 않거나, 입력 유효성검사를 통해 안전을 검증하지 않는다면 XSS에 취약하다. 적절한 출력 값 이스케이핑 또는 유효성 검사 없이 브라우저에서 입력이 되면 활성화된 콘텐츠로 처리된다. 동적으로 페이지를 업데이트하는데 Ajax를 사용하는 경우 안전한 JavaScript APIs 를 사용하고 있는가? 확인하고 안전하지 않은 자바스크립트 API에 대한 인코딩 또는 유효성 검사도 해야 한다. 자동화된 도구는 자동적으로 일부 XSS 문제점을 발견할 수 있다. 하지만, 각각의 애플리케이션은 다른 방식으로 출력 페이지를 구축하고, 자바스크립트 등과 같은 다른 브라우저 인터프리터를 사용하기 때문에 자동 탐지는 어렵다. 자동 접근 방식뿐만 아니라 전체적인 범위에서 수동으로 코드를 검토하고, 침투시험이 필요하다. 자동화된 도구로 Ajax와 같은 웹 2.0 기술에서 XSS를 탐지하는 것은 더욱 어렵다.

3.4 A4 보안 취약성 확인 방법

취약한 직접 객체 참조는 개발자가 파일, 디렉토리, 데이터베이스 키와 같은 내부 구현 객체를 참조하는 것을 노출시킬 때 발생한다. 접근 통제를 통한 확인이나 다른 보호수단이 없다면, 공격자는 노출된 참조를 조작하여 허가 받지 않은 데이터에 접근할 수 있다.

확인방법은 모든 객체 참조 시 적절한 보안을 취하고 있는지 검증하는 것이다.

- 제한된 자원에 직접적으로 참조하는 경우, 애플리케이션에서 사용자가 요청한 정확한 자원에 접근할 수 있도록 승인이 되었는지 검증하지 않는가?
- 만약 이러한 참조가 간접적인 참조라면, 직접 참조에 대한 매핑은 기존 사용자에게 허용된 값으로만 제한하지 않는가?

애플리케이션 코드 검토를 통해 각각의 접근이 안전하게 실행되는 지를 검증할 수 있다. 시험을 하면 직접적인 객체 참조를 확인하거나 안전한지 아닌 지 확인할 수 있다. 자동화된 도구는 보호할 것이 무엇인지 또는 무엇이 안전하고 불안전한지 알지 못하기 때문에 이 같은 결함을 찾을 수 없다

3.5 A5 보안 취약성 확인 방법

보안 설정 오류는 기본으로 제공되는 값은 종종 안전하지 않기 때문에 보안 설정은 정의, 구현 및 유지되어야 한다. 또한 소프트웨어는 최신의 상태로 유지해야 한다.

확인방법은 애플리케이션의 스택 모든 부분에 보안 강화가 되고 있는가? 예를 들자면 다음과 같다.

- 운영체제, 웹/앱 서버, DBMS, 애플리케이션이나 코드 라이브러리들도 포함해서 보안 업데이트 기간이 지난 소프트웨어가 있지는 않은가?

- 불필요한 기능이 활성화되어 있거나 설치되어 있지는 않은가?
- 기본 계정들 및 관련된 패스워드들이 아직 활성화되어 있고 한 번도 변경을 하지 않았는가?
- 에러 처리 부분이 스택 추적 가능한 수준으로 자세하거나 사용자에게 보여 지는 에러 메시지가 너무 지나치게 자세하지는 않은가?
- 개발 중인 프레임워크의 보안 설정과 라이브러리들의 보안 설정을 보증하실 수 있는가? 합의된 반복 가능한 애플리케이션 보안 설정 프로세스가 없다면 시스템은 높은 위험을 가지고 있는 것이다.

3.6 A6 보안 취약성 확인 방법

민감 데이터 노출은 많은 웹 애플리케이션들이 신용카드, 개인 식별 정보 및 인증 정보와 같은 중요한 데이터를 제대로 보호하지 않는다. 공격자는 신용카드 사기, 신분 도용 또는 다른 범죄를 수행하는 등 약하게 보호된 데이터를 훔치거나 변경할 수 있다. 중요 데이터가 저장 또는 전송 중이거나 브라우저와 교환하는 경우 특별히 주의하여야 하며, 암호화와 같은 보호조치를 취해야 한다.

확인방법은 어떤 정보가 추가적인 보호가 필요한 민감한 데이터인지 결정하는 것이다. 예를 들어, 패스워드, 신용카드 정보, 건강기록, 개인정보는 반드시 보호되어야 하는 것들이다. 이러한 정보에 대해서 아래와 같은 내용을 확인한다.

- 민감한 정보들이 사용 중인 저장 공간이나 백업 공간에 저장될 때 암호화되지 않은 긴 문자로 저장되지는 않는가?
- 이러한 정보들이 내부나 외부에 암호화되지 않고 전송되는가? 인터넷 트래픽은 특히 위험하다.
- 오래되었거나 취약한 암호 알고리즘을 사용하지는 않는가?
- 취약한 암호키가 생성되었거나 적절한 키 관리는 이루어지는가? 또는 순환이 누락되지는 않는가?
- 브라우저 보안지침이나 헤더가 민감한 데이터가 제공되었거나 브라우저에 보내졌을 때 놓치지는 않는가?

3,7 A7 보안 취약성 확인 방법

기능수준의 접근통제 누락은 UI(User Interface)에 해당 기능을 보이게 하기 전에 기능 수준의 접근권한을 확인한다. 그러나 애플리케이션은 각 기능에 접근하는 서버에 동일한 접근통제 검사를 수행한다. 요청에 대해 적절히 확인하지 않을 경우 공격자는 적절한 권한 없이 기능에 접근하기 위한 요청을 위조할 수 있다.

확인방법은 모든 애플리케이션 기능을 검증하는 것이다.

- 인증되지 않은 기능이 UI에서 제공되고 있지 않은가?
- 서버 쪽 인증과 인증 체크가 이루어 지지 않았는가?
- 서버 쪽 체크가 공격자가 제공하는 정보에 의존하고 있지 않은가?

프록시를 이용해서 애플리케이션이 특별한 역할이 있는지 확인하고, 그 다음 권한이 낮은 제한된 페이지에 다시 방문해 보아라. 이때 서버 응답이 똑같다면 취약점이 존재할 가능성이 크다. 프록시는 이와 같은 분석 기능을 지원하기도 한다. 또한 접근 통제를 구현한 코드를 확인할 수 있다. 코드와 인가된 인증 패턴을 통해 단일 권한 요청(single privileged request)을 시도할 수 있다. 추적되지 않는 패턴이 어디에 있는지 찾기 위해 코드기반(code-based)를 검색해 볼 수 있다. 자동화된 도구는 이러한 문제들을 못 찾을 가능성이 있다.

3.8 A8 보안 취약성 확인 방법

크로스 사이트 요청 변조 (CSRF)는 로그온 된 피해자의 취약한 웹 애플리케이션에 피해자의 세션 쿠키와 기타 다른 인증정보를 자동으로 포함하여 위조된 HTTP 요청을 강제로 보내도록 하는 것이다. 이것은 공격자가 취약한 애플리케이션이 피해자로부터의 정당한 요청이라고 오해할 수 있는 요청들을 강제로 만들 수 있다.

확인방법은 링크 및 폼들이 예측할 수 없는 CSRF 토큰이 누락되었는지 봐야 한다. 그러한 토큰이 없다면, 공격자들은 악의적인 요청들을 위조할 수 있다. 대체 방어책은 사용자에게 요청을 제출하는 의도를

증명하거나, 재인증 또는 진짜 사용자라는 일부 다른 증거를 요구하는 것이다. 상태를 변경하는 함수는 가장 중요한 CSRF 공격목표가 되기 때문에 이러한 함수를 호출하는 링크나 폼에 집중해야 한다. 그리고 다단계 처리기능은 기본적인 방어책이 없기 때문에 확인해야 한다. 공격자들은 다양한 기술 또는 아마도 자바스크립트를 사용하여 연속적으로 요청들을 쉽게 위조할 수 있다. 브라우저에서 자동적으로 보내는 세션 쿠키, 출발지 IP 주소, 그리고 다른 정보들이 CSRF에 대한 방어책을 제공하지 않는지 확인해야 한다. 왜냐하면 이러한 정보도 위조된 요청에 포함되어 있기 때문이다.OWASP의 CSRF Tester 도구는 CSRF 결함 위험을 시연하기 위한 시험 케이스를 생성하는데 도움이 될 수 있다.

3.9 A9 보안 취약성 확인 방법

알려진 취약점이 있는 컴포넌트 사용은 컴포넌트 대부분 항상 전체 권한으로 실행된다. 이러한 취약한 컴포넌트를 악용하여 공격하는 경우 심각한 데이터 손실이 발생하거나 서버가 장악된다. 알려진 취약점이 있는 컴포넌트를 사용하는 애플리케이션은 애플리케이션 방어 체계를 손상하거나, 공격 가능한 범위를 활성화하는 등의 영향을 미친다.

확인방법은 여러분이 사용하는 취약한 컴포넌트나 라이브러리를 쉽게 찾아낼 수 있지만, 불행히도 상업용 또는 오픈 소스 소프트웨어에 대한 취약점 보고서는 항상 취약한 버전이 정확히 어떤 것인지에 대해 기대하는 수준만큼, 그리고 찾기 쉽도록 명시하지 않는다. 또한 모든 라이브러리가 이해하기 쉬운 버전관리 시스템을 사용하지도 않는다. 가장 심각한 것은, CVE나 NVD 에서 쉽게 검색이 가능한 취약점들조차도 모두 중앙 관리 조직에 보고되지 않는다는 것이다. 취약한지 결정하기 위해서는 이러한 데이터베이스를 검색하는 것뿐만 아니라 프로젝트 메일링 리스트, 그리고 취약점과 관련된 발표내용도 잘 파악해야 한다. 만일 사용 중인 컴포넌트 중 어느 하나라도 취약하다면, 사용 중인 코드를 검토하여 실제로 해당 컴포넌트의 취약한 부분을 사용하고 있는지, 해당 결함이 영향을 미칠 수 있는지를 확인해야 한다

3.10 A10 보안 취약성 확인 방법

검증되지 않은 리다이렉트 및 포워드는 웹 애플리케이션은 종종 사용자들을 다른 페이지로 리다이렉트하거나 포워드하고, 대상 페이지를 결정하기 위해 신뢰할 수 없는 데이터를 사용한다. 적절한 검증 절차가 없으면 공격자는 피해자를 피싱 또는 악성코드 사이트로 리다이렉트 하거나 승인되지 않은 페이지에 접근하도록 전달할 수 있다.

확인방법은 다음과 같다.

- 코드에 사용된 모든 리다이렉트 혹은 포워드(.NET에서는 transfer)를 점검하라. 이 코드들의 파라미터 값에 URL이 포함되어 있고, 또 그 목적지 URL이 허용된 페이지 목록에 없다면, 취약점이 존재한다.)
- spider(웹 사이트 정보 수집 도구)를 이용하여 해당 사이트가 리다이렉트(HTTP 응답 코드 300-307, 보통은 302)를 발생시키는지 점검해야 한다. 리다이렉트보다 앞서 나오는 파라미터를 조사하여 목적지 URL인지, 혹은 그 일부인지 분석해야 한다. 만약 URL 혹은 그 일부라고 판단될 경우 목적지 URL을 변경하고, 변경된 목적지로 리다이렉트가 발생하는지 관찰해야 한다.
- 만약 코드를 입수할 수 없다면 모든 파라미터를 조사해서 리다이렉트 혹은 포워드 목적지 주소의 일부인지 아닌지 조사하고, 이 경우 어떤 행동을 하는지 시험해야 한다.

4. 보안취약점 검증 및 성능검사

본 논문에서 제시한 웹 애플리케이션에 대한 보안 취약점에 대해서 개선방법으로 제시한 방법을 적용에 대한 웹 애플리케이션 보안 취약점 검증과 성능검사 2가지 방법으로 시도하였다. 첫 번째는 웹 애플리케이션 보안 취약점 검증을 통해 점검된 보안성 강화 여부를 검토하였다. 두 번째는 웹 애플리케이션 보안 취약점이 개선 후 성능 검사를 검토하였다[16].

4.1 보안 취약점 검증

보안취약점 검증방법으로는 모의해킹 및 소스코드

분석은 해커 수동점검 및 분석도구를 사용해서 분석 및 점검을 하고, 애플리케이션 분석은 검사목록을 작성하여 해당 목록을 점검하는 것이다. 본 논문에서는 보안 취약점 점검도구와 웹 애플리케이션에 대한 취약점 점검 목록을 소개한다. 보안 취약점 검검도구는 <표 1>과 같이 OWASP TOP 10 대 웹 애플리케이션 보안 위험의 8개에 해당하는 보안 취약점을 검사는 도구이다[17].

<표 1> 보안 취약점 및 점검도구목록

RISK	TOOL
A1 Injection	SQL Inject Me
A2 Broken Authentication and Session Management	HackBar
A3 Cross-Site Scripting (XSS)	ZAP
A4 Insecure Direct Object References	Burp
A5 Security Misconfiguration	Watobo
A6 Sensitive Data Exposure	Nikto/Wikto
A9 Using Components with Known Vulnerabilities	Watobo
A10 Unvalidated Redirects and Forwards	Watcher

보안취약점에 대한 검증을 하기 위해서 <표 2>와 같은 웹 애플리케이션에 대한 취약점 점검 목록을 이용한다[18][19].

<표 2> 웹 애플리케이션 보안취약점점검목록

SQL Injection 보안점검 항목				
연번	점검항목	O	X	비고
1	로그인 폼에 대해 점검하였는가?			
2	게시판 글 조회 란에 대해 점검하였는가?			
3	게시판 URL 조작에 대해 점검하였는가?			
4	회원가입 페이지 ID 조회 란에 대해 점검하였는가?			
5	우편번호 조회 란에 대해 점검하였는가?			
6	확장 프로시져 기능에 대해 점검하였는가?			
XSS 보안점검 항목				
1	게시판의 입력란 (제목, 작성자, 메일주소, 글 입력란)에 대해 점검하였는가?			
2	홈페이지의 조회 란에 대해 점검하였는가?			
3	홈페이지 URL 조작에 대해 점검하였는가?			
파일 업로드 보안점검 항목				
1	게시판에 업로드 되는 파일의 확장자를 체크하는가?			
2	변경된 첨부파일의 확장자를 인식할 수 있는가?			
쿠키 변조 보안점검 항목				
1	쿠키 내의 id 값을 변경 시 인식할 수 있는가?			
2	쿠키 내의 코드 값 변경 시 인식할 수 있는가?			
다운로드 취약점 보안점검 항목				
1	다운로드 URL에 대한 유효성 여부를 체크하였는가?			
2	웹 서버에서 다운로드 가능한 확장자를 등록하였는가?			
3	디렉토리 리스팅이 발생하지 않도록 설정하였는가?			

4.2 보안취약점 개선후 부하성능검사

보안취약점 개선 후 부하성능 검사를 하기 위해 보안이 추가되기 전과 후에 대해 최대한 자원을 공유하고 재사용이 가능한지에 대해 부하성능 테스트를 진행하였다. 본 논문에서는 검증을 위해 활용한 자료는 다음과 같은 사업의 특성을 가진 웹 사이트를 선정하였다.

- 사업명 : 80여개 웹 사이트 구축 및 개발
- 웹서버 : IBM서버 2대
- DB서버 : IBM서버 1대
- L4스위치를 통한 로드밸런스

4.3 개발 시스템에 대한 부하성능 평가

개발된 시스템의 보안 취약점 분석과 이의 보완을 위해 OWASP TOP 10의 보안항목을 기준으로 하되, 전문적인 소스코드 취약점을 분석하여 주는 도구를 사용하였다. 또한 통합엔진의 성능을 평가하기 위해 보안이 추가되기 전과 후에 대해 최대한 자원을 공유하고 재사용이 가능한지에 대해 부하성능 테스트를 진행하였다. <표 3>은 테스트 부하 수준에 대한 정의를 나타내었다.

<표 3> 테스트 부하 수준에 대한 기준

부하수준	USER수 (명)	웹 서버 CPU 사용량	비고
저부하	30	40%	
중부하	60	70~80%	
고부하	90	90% 이상	

<표 4> 처리량 및 응답시간 비교

구분		평균 (Pages/sec)	평균 (Hits/sec)	평균 페이지 응답시간	평균 요청 응답시간
저부하	추가전	8.8	12.4	0.263s	0.186s
	추가후	8.8	12.4	0.261s	0.184s
	증감율	0%	0%	-0.80%	-1.10%
중부하	추가전	16.6	23.5	0.446s	0.316s
	추가후	16.6	23.5	0.449s	0.318s
	증감율	0%	0%	0.70%	0.60%
고부하	추가전	19	27.1	1.53s	1.08s
	추가후	19.2	27.2	1.51s	1.07s
	증감율	1.10%	0.40%	-1.30%	-0.90%

<표 4, 5, 6>은 개발된 웹 사이트에 대해 보안 취약점 개선을 적용하기 전과 후의 성능테스트 결과를 보이고 있다. 테스트는 각각 저부하, 중부하, 고부하의 단계별로 진행 하였다.

<표 4>에서 Pages/sec는 웹 서버에서 초당 사용자 페이지 회신한 처리량이고, Hits/sec는 웹 서버에서 초당 HTTP Requests를 회신한 처리량이며, 페이지 응답시간은 웹 서버에서 사용자 페이지에 대한 처리한 시간이다. 그리고 요청 응답시간은 웹 서버에서 HTTP Request 건당 처리한 시간이다.

<표 5> 웹 서버 테스트 결과

구분		평균 CPU User(%)	평균 CPU System(%)	평균 CPU Idle(%)
저부하	추가전	30.3	2.41	67
	추가후	29.9	2.39	67.4
	증감율(%)	-1.10%	-1%	0.60%
중부하	추가전	68.7	4.86	26.3
	추가후	69	4.91	26.1
	증감율(%)	0.40%	1%	-0.80%
고부하	추가전	88.6	6.55	4.92
	추가후	88.8	6.47	4.79
	증감율(%)	0.20%	-1.30%	-2.70%

<표 6> DB 서버 테스트 결과

구분		평균 CPU User(%)	평균 CPU System(%)	평균 CPU Idle(%)
저부하	적용전	5.52	4.25	87.3
	적용후	5.68	4.3	87.1
	증감율(%)	3%	1.20%	-0.20%
중부하	적용전	10.77	8.35	78.3
	적용후	11.4	8.88	77.2
	증감율(%)	5.70%	6.30%	-1.40%
고부하	적용전	13.2	10.11	74.2
	적용후	13.1	10.4	74.1
	증감율(%)	-1.20%	2.90%	-0.10%

부하성능 테스트 결과는 필터링 적용에 따른 처리량 및 응답시간 변화가 1% 미만으로 성능 변화가 거의 없는 것으로 추정된다. 필터링 적용에 따른 서버의 CPU 사용량 변화가 1% 수준으로 영향도가 거의 없는 것으로 추정된다.

4.4 검증결과

보안성취약점 개선방법을 적용하여 보안을 강화하였을 때 성능에서도 많은 차이를 보이지 않고 약간의 차이를 보였음을 알 수 있었다. 따라서 보안성취약점 개선방법을 적용하기 전과 적용했을 때의 성능에서 별 차이가 없기 때문에 보안 취약점 개선방법을 적용하는 것이 안전한 사이트를 만들기 위해서 필요하다.

5. 결 론

최근 웹 사이트의 보안이 웹 애플리케이션 제작 시 취약점을 이용한 개인정보의 노출과 해킹으로 심각한 사회문제로까지 대두되어 안전한 웹 사이트 제작방법이 절실히 요구되고 있다. 따라서 본 논문에서 분석하였던 여러 가지 웹 사이트 보안 문제를 해결하기 위한 방안이 지속적으로 연구되어져야 할 것이다.

본 논문은 개인정보의 노출과 해킹으로부터 안전한 사이트를 만드는 방법에 대해 연구하였고, 문헌자료와 각종 보안대책 자료를 이용하여 안전한 웹 사이트 제작방법을 제시하였다.

본 논문은 웹 사이트 제작 시에 필요한 개발적인 보안취약점 및 개선방법의 연구만으로 미흡한 점이

있다. 향후 연구에서는 시스템해킹, 네트워크해킹, 모바일 웹 사이트 보안 문제해결, 웹 사이트 보안도구 즉, 휘슬, 캐슬, 공개 웹 방화벽을 다 적용한다면 해킹에 더 안전한 웹 사이트가 될 것이다.

참고문헌

[1]. Gartner, Now is the time for security at Application Level, 2006.12

[2] 행정안전부, 소프트웨어 개발 보안 가이드, 2011.06

[3]. http://ko.wikipedia.org/wiki/보안

[4] 박상노, 클라우드 컴퓨팅 서비스의 보안 전략 방안에 관한 연구, 배재대학교 대학원, 2013.06

[5] http://ko.wikipedia.org/wiki/클라우드_컴퓨팅

[6] Jasti, Amarnath, et al. "Security in multi-tenancy cloud.", 2010

[7] SaaS 플랫폼 기술 및 개발 동향, ETRI, 전자통신동향분석 제26권 제5호, 2011.10

[8] SaaS 기술개발동향, ETRI, 전자통신동향분석 제24권 제4호 2009.08

[9] 차영태, 클라우드 컴퓨팅 보안 기술동향과 산업전망 JULY 2012 VOL 12-6, KEIT, 2012.07

[10] 류미현, 클라우드 컴퓨팅에서 샌드박스 기반의 멀티테넌시 데이터 보안, 동국대학교 국제정보대학원, 2014

[11] 차영태, 클라우드 컴퓨팅 보안 기술동향과 산업전망, 한국산업기술평가관리원 PD ISSUE REPORT July 2012 vol. 12-6.

[12] http://en.wikipedia.org/wiki/Multitenancy

[13] http://ko.wikipedia.org/wiki/OWASP

[14] http://www.owasp.or.kr, OWASP Top 10 2013 Final Korean, 2013

[15] 행정안전부, 홈페이지 개인정보 노출방지 가이드라인, 2012.07

[16] 행정안전부, 공공기관 홈페이지 개인정보 노출방지 가이드라인, 2011

[17] KISA, 국내 클라우드 서비스 보안 취약점 점검

[18].http://resources.infosecinstitute.com/owasp-top-10-tools-and-tactics/

[19] KISA, 웹서버구축보안점검 안내서, 2010.01

[저 자 소 개]

이 도 현 (Do Hyeon Lee)

2001년 2월 한양대학교 전자전기공학부 학사
2003년 8월 한양대학교 전자통신전파공학과 석사
2011년 2월 한양대학교 전자통신컴퓨터공학과 박사
2011년 4월 ~ 2014년 2월 남서울대학교 IT융합기술사업단 연구교수
2014년 3월 ~ 2015년 7월 ㈜유아이넷 책임연구원
2015년 8월 ~ 현재 (주)신성씨엔티 책임연구원

email : dohyeon@gmail.com

이 종 욱 (Jong_Woock Lee)

1984년 2월 서울시립대학 전자공학 학사
1993년 2월 광운대학교 전자계산기공학 석사
2009년 2월 선문대학교 컴퓨터공학 박사
1995년 7월 ~ 현재 한국폴리텍대학 모바일정보통신과 교수

Email :: jwlee@kopo.ac.kr

김 점 구 (Jeom Goo Kim)

1990년 2월 광운대학교 전자계산학과 이학사
1997년 8월 광운대학교 전자계산학과 석사
2000년 8월 한남대학교 컴퓨터공학 박사
1999년 3월~ 현재 남서울대학교 컴퓨터학과 교수
IT융합연구소장

email : jgoo@nsu.ac.kr