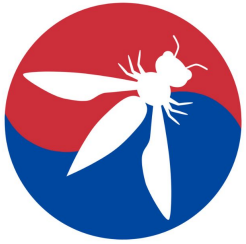
OWASP Top 10 2025:
변한 것 그리고 변하지 않은 것

OWASP 서울 챕터



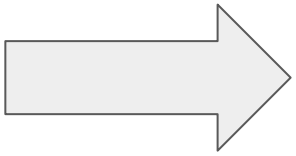
OWASP Top 10의 역사

OWASP Top 10은 웹 취약점 리스트를 공유하기 위하여 2003년에 시작되었으며, 2003년 첫 릴리스 이후, 2025년 버전은 8번째 업데이트이며, 4년마다 업데이트되며 웹 보안 환경의 변화를 반영합니다.

*OWASP는 웹 취약점을 공유하기 위해 출발했지만, 이제는 웹을 넘어 다양한 보안 주제를 다루고 있기 때문에 ‘Open World’로 확장되었습니다.

목적

OWASP Top 10은 웹 애플리케이션에 대한 가장 중요한 10가지 위험을 다루며, 취약점의 순위가 아닌 **위험(Risks)의 순위**입니다. 또한, 데이터 기반으로 이루어진 인식 재고를 위한 문서이며, 표준이나 규정 준수 체크리스트가 아닌, 조직이 보안 위험에 우선순위를 부여하도록 돕는 역할을 합니다.

단순 취약점 순위  **리스크 순위**

커뮤니티 주도

OWASP TOP 10은 커뮤니티 주도적으로 만들어지며, 회사 및 조직들로부터 수백만 건의 기록과 수백 명의 설문 응답자로부터 받은 데이터를 기반으로 구축됩니다.

2025년 목록을 결정하는 과정은 데이터 수집(14~16개월 소요), 데이터 정규화, NVD(National Vulnerability Database)를 통한 CVE-CWE 연결, CVSS 점수를 사용한 익스플로잇 및 영향도 정규화, 커뮤니티 설문조사 실행, 그리고 데이터와 설문조사를 통합하여 새로운 Top 10을 결정하는 단계를 포함합니다.

* 2025년 Top 10 프로젝트의 공동 리더 : Brian Glas, Torsten Gigler, Neil Smithline, Andrew van der Stock, Tanya Janca

* 참여 조직: Accenture (Prague), Anonymous (multiple), Bugcrowd, Contrast Security, CryptoNet Labs, Intuitor SoftTech Services, Orca Security, Probley, Semgrep, Sonar, usd AG, Veracode, Wallarm

2025의 주된 변화

2025년 버전의 주된 변화로 현대 소프트웨어 개발의 변화를 반영하는 것입니다. 이는 취약점 중심의 시각에서 리스크(risk) 관리와 복원력(resilience) 중심의 모던 애플리케이션 보안의 위주로 이동하는 것을 보여줍니다.

증상에서 근본 원인 초점

증상(예시: 민감 정보 노출)을 주목하기보다 근본 원인(예시: 암호화 실패)을 식별하는 데 중점을 둡니다.

시스템적 약점 강조

클라우드 네이티브 패턴에서 발생하는 문제점, 오픈 소스 생태계 및 자동화된 CI/CD 파이프라인의 도입으로 인해 발생하는 문제점과 같이 시스템적 약점에 중점을 둡니다.

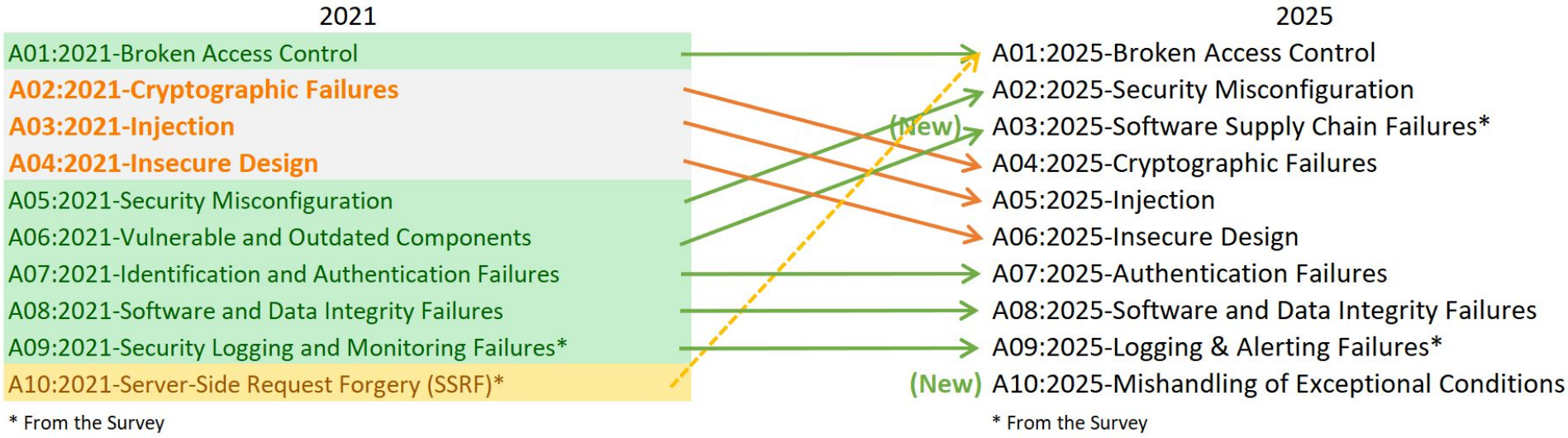
SDLC 전반의 보안 강조

애플리케이션 보안이 단순히 소스 코드를 넘어 코드가 빌드되고, 종속성을 관리하고, 파이프라인이 작동하고, 배포 환경이 구성되는 방식까지 포함한다는 점을 강조합니다.

2003	2004	2007	2010
A1-Unvalidated Parameters	A1-Unvalidated Input	A1-Cross Site Scripting (XSS)	A1-Injection
A2-Broken Access Control	A2-Broken Access Control	A2-Injection Flaws	A2-Cross-Site Scripting (XSS)
A3-Broken Account and Session Management	A3-Broken Authentication and Session Management	A3-Malicious File Execution	A3-Broken Authentication and Session Management
A4-Cross Site Scripting (XSS) Flaws	A4-Cross Site Scripting (XSS) Flaws	A4-Insecure Direct Object Reference	A4-Insecure Direct Object References
A5-Buffer Overflows	A5-Buffer Overflows	A5-Cross Site Request Forgery (CSRF)	A5-Cross-Site Request Forgery (CSRF)
A6-Command Injection Flaws	A6-Injection Flaws	A6-Information Leakage and Improper Error Handling	A6-Security Misconfiguration
A7-Error Handling Problems	A7-Improper Error Handling	A7-Broken Authentication and Session Management	A7-Insecure Cryptographic Storage
A8-Insecure Use of Cryptography	A8-Insecure Storage	A8-Insecure Cryptographic Storage	A8-Failure to Restrict URL Access
A9-Remote Administration Flaws	A9-Denial of Service	A9-Insecure Communications	A9-Insufficient Transport Layer Protection
A10-Web and Application Server Misconfiguration	A10-Insecure Configuration Management	A10-Failure to Restrict URL Access	A10-Unvalidated Redirects and Forwards

2013	2017	2021	2025
A1-Injection	A1:2017-Injection	A01:2021-Broken Access Control	A01:2025-Broken Access Control
A2-Broken Authentication and Session Management	A2:2017-Broken Authentication	A02:2021-Cryptographic Failures	A02:2025-Security Misconfiguration
A3-Cross-Site Scripting (XSS)	A3:2017-Sensitive Data Exposure	A03:2021-Injection	A03:2025-Software Supply Chain Failures*
A4-Insecure Direct Object References	A4:2017-XML External Entities (XXE)	A04:2021-Insecure Design	A04:2025-Cryptographic Failures
A5-Security Misconfiguration	A5:2017-Broken Access Control	A05:2021-Security Misconfiguration	A05:2025-Injection
A6-Sensitive Data Exposure	A6:2017-Security Misconfiguration	A06:2021-Vulnerable and Outdated Components	A06:2025-Insecure Design
A7-Missing Function Level Access Control	A7:2017-Cross-Site Scripting (XSS)	A07:2021-Identification and Authentication Failures	A07:2025-Authentication Failures
A8-Cross-Site Request Forgery (CSRF)	A8:2017-Insecure Deserialization	A08:2021-Software and Data Integrity Failures	A08:2025-Software or Data Integrity Failures
A9-Using Components with Known Vulnerabilities	A9:2017-Using Components with Known Vulnerabilitie	A09:2021-Security Logging and Monitoring Failures*	A09:2025-Logging & Alerting Failures*
A10-Unvalidated Redirects and Forwards	A10:2017-Insufficient Logging & Monitoring	A10:2021-Server-Side Request Forgery (SSRF)*	A10:2025-Mishandling of Exceptional Conditions

2025의 주된 변화


2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey
2025
A01:2025-Broken Access Control
A02:2025-Security Misconfiguration
(New) A03:2025-Software Supply Chain Failures*
A04:2025-Cryptographic Failures
A05:2025-Injection
A06:2025-Insecure Design
A07:2025-Authentication Failures
A08:2025-Software and Data Integrity Failures
A09:2025-Logging & Alerting Failures*
(New) A10:2025-Mishandling of Exceptional Conditions
* From the Survey

A01: 접근 통제 실패 (Broken Access Control)

BAC는 애플리케이션이 사용자가 **허용된 권한을 넘어서**는 행동을 막지 못할 때 발생합니다. 이로 인해 공격자가 권한을 우회하거나, 무단으로 정보 조회·수정·삭제 또는 비즈니스 기능을 수행할 수 있습니다.

 # 순위가 변하지 않은 이유

- 맥락을 잘 이해해야지만 찾을 수 있는 까다로운 문제.
- SAST(정적분석도구), DAST(동적분석도구)로 찾기 어려움. (AI가 과연 이 문제를 해결할 것 인가..!?)
    - 현대 웹 애플리케이션에서 가장 취약하고 테스트하기 어려운 구성 요소

 # 대표적인 취약점

- IDOR(Insecure Direct Object Reference), 권한 상승 등
- SSRF(Server-Side Request Forgery) - 2021의 A10이였던 SSRF가 BAC에 편입입

 # 대응 방안

- 권한 검증 미들웨어를 통한 역할 기반(RBAC)과 같은 견고한 권한 검증
- 순차적으로 증가하지 않는 유니크 키 사용
- 권한 테스트 케이스 작성
- Deny by default를 통해서 실수를 예방(기본적으로 deny, 권한이 있으면 수행 if True: action(); fail())

A02: 보안 오설정 (Security Misconfiguration) ▲3

보안 오설정은 시스템, 애플리케이션, 클라우드 서비스, 컨테이너 또는 IaC 템플릿 등이 **보안 관점에서 잘못 설정**되거나, 안전하지 않은 기본값을 사용하여 배포될 때 발생하는 위험입니다.

순위가 오른 이유

- 클라우드, SaaS 도구, 외부 라이브러리 등 애플리케이션의 개발 환경의 복잡도가 높아진 이유로 코드 보안 자체는 개선되고 있더라도 전체 환경의 보안의 복잡성이 증가했다는 점을 시사

대표적인 취약점

- 클라우드 설정 오류(S3 버킷 노출, IAM 과도한 권한 부여 등)
- CORS(Cross-Origin Resource Sharing) 오설정
- 잘못 구성된 HTTP 보안 헤더(예: Content Security Policy)
- 기본 비밀번호 사용
- XXE 외부 엔티티 허용

대응 방안

- 기본이 잘 설정된 OS 이미지, 컨테이너 이미지, IaC 템플릿을 만들어 사용 및 템플릿 보안 스캔
- 미들웨어를 통한 보안 헤더 일괄 적용

New!

A03: 소프트웨어 공급망 보안 실패 (Software Supply Chain Failures) ▲3

2021년의 A06 취약하고 오래된 구성 요소를 대체하는 새로운 항목. 단순히 오래된 라이브러리를 관리리하는 것을 넘어서 사용하는 공급망에서의 문제 및 공급망 관리의 보안까지 포함하는 항목으로 변화했습니다.

순위가 오른 이유

- 데이터 상에서는 높은 점수를 받지 못했지만, 커뮤니티 설문조사에서 **1위**를 차지함.
- 잘 방어하기에 소요되는 리소스가 많이 듬.

대표적인 취약점

- 외부 종속성에서 발생하는 취약점(예: React2Shell)
- 빌드 시스템, 서명 인서 해킹
- npm이나 PyPI와 같은 패키지 저장소 해킹

대응 방안

- SCA(Software Composition Analysis) 도구 사용 및 SBOM(Software Bill of Materials)을 구축하여 종속성을 관리하여 리스크 발생시 빠른 대응
- 빌드 파이프라인의 접근 통제 및 외부 공급망(주로 레파지토리)의 접근 권한 관리 및 MFA
- 사용하지 않는 종속성 제거, 종속성 선택시 신중하게 정

A04: 암호학적 실패 (Cryptographic Failures) ▼2

암호학적 결함은 민감한 데이터를 노출시킬 수 있는 암호화 부재, 암호화 강도가 낮은 암호화, 암호화 키 유출 및 하드코딩 된 암호화 키와 같은 문제를 다룹니다.

순위가 낮아진 이유

- 여전히 중요하지만, 다른 위협이 치고 올라 옴...!

대표적인 취약점

- 약한 알고리즘 사용
- 부적절한 암호화 키 관리(하드코딩, 키 교체 주기, 키 접근 제어)
- 데이터 암호화 미흡

대응 방안

- AES-256, SHA-256, TLS 1.3과 같이 강력하며 잘 검증된 암호화 표준을 사용.
- 암호화 키를 하드코딩 하지 않고, 통제된 저장소에 안전한 방식으로 저장해야 하며 자동 교체 기능을 수행.
- 평문으로 저장될 필요가 없는 경우 패스워드 전용 해싱 알고리즘(bcrypt, scrypt, Argon2) 사용.

A05: 인젝션 (Injection) ▼2

신뢰할 수 없는 데이터가 **쿼리, 명령어 또는 템플릿**으로 유입될 때 발생하며,
애플리케이션이 이를 데이터가 아닌 실행 가능한 코드나 명령어의 일부로 처리하는 위험.

순위가 낮아진 이유

- 정적분석도구(SAST)에서 발견이 비교적 쉬움.
- 프레임워크, 라이브러리에서 안전한 코딩을 위한 기능을 기본으로 제공 함
- 그럼에도 발생시 파급력이 높기 때문에 여전히 5위에 위치. (2003릴리즈부터 2017년까지 1위였음)

대표적인 취약점

- SQL Injection, Command Injection.
- XSS(Cross Site Scripting)
- 프롬프트 Injection

대응 방안

- 쿼리의 경우 prepared statements(지원하는 경우)이나 parameterized queries 사용
- SAST/DAST 도구 사용 - 개발 과정내에서 탐지될 수 록
- 브라우저가 지원하는 보안 기능(CSP 등) 활용

A06: 안전하지 않은 설계 (Insecure Design) ▼2

코딩 단계가 아닌 **설계 단계에서 보안 제어가 누락**되거나 비효율적일 때 발생하는 위험입니다.

순위가 낮아진 이유

- 다른 위협이 치고 올라옴…
- 여전히 상위권에 속하는 이유는, 한번 만들면 바꾸기 어려운 구조기 때문

대표적인 취약점

- 비즈니스 로직 취약점 류(단체 예매 창구로 예매시 할인이 있으나 1명도 예약이 가능함)
- 비밀번호 찾기시 질문/답변 기능 제공
- 봇의 자동화 공격 대응 미흡

대응 방안

- 위협 모델링 수행
- 개발 이전 디자인 리뷰
- 개발팀과 함께 보안 적으로 안전한 공통 모듈을 만들어 해당 기능을 사용하도록 가이드
- Secure by Design - 보안을 기본 요구사항으로 취

A07: 인증 실패 (Authentication Failures)

공격자가 **유효하지 않거나 부정확한 사용자**를 정상적인 요청으로 인식하도록 시스템을 속일 수 있게 만드는 위험입니다.

A01(BAC) 과 다른점은?

- A01은 인가(authz)에 초점을 둔다면 A07은 인증(authn)에 초점을 둠.

대표적인 취약점

- 약한 비밀번호 정책
- 크리덴셜 스터핑 및 무차별 대입 공격
- MFA 기능 누락/MFA 사용안함
- 세션 관리 오류(로그아웃 후에도 세션 토큰 유지, 세션 키가 URL에 노출되는 문제 등)

대응 방안

- 강력한 비밀번호 정책 및 보호 - 유출되었거나 약한 비밀번호를 사용하지 못하게 제한
- 표준 인증 라이브러리 사용(직접 인증 로직을 만들지 않고 검증된 로직 활용)
- 안전한 세션 관리

A08: 소프트웨어 또는 데이터 무결성 실패 (Software or Data Integrity Failures)

애플리케이션에서 데이터 유효성, 출처 또는 무결성이 확인되지 않은 채 처리되거나 사용될 때 발생하는 보안 위험입니다.

A03(소프트웨어 공급망 보안 실패) 과 다른점은?

- A08은 외부의 종속성 자체가 안전하더라도, 애플리케이션 환경 내부에서 이루어지는 무결성 검증 실패에 초점.

대표적인 취약점

- 안전하지 않은 역직렬화(React2shell의 루트커즈)
- CDN 무결성 유효성 검증 미흡
- 업데이트시 디지털 서명 검증 미흡
- 신뢰하지 않는 종속성 저장소 사용

대응 방안

- 신뢰할 수 있는 종속성 저장소(npm이나 Maven)만 사용, 위험도가 높은 경우 자체 저장소 사용 고려
- 업데이트시 서명을 검증
- 역직렬화를 수행하는 경우, 신뢰되는 클라이언트에서 전송되었는지 검증

A09: 로깅 및 알람 실패 (Logging & Alerting Failures)

적절한 보안 로그를 수집하지 않거나, 이를 모니터링하지 않았거나, 또는 알람 경고에 신속하게 대응하지 못할 때 발생하는 위험입니다.

대표적인 취약점

- 필요한 정보가 없거나 불충분한 로그
- 로그 무결성 실패
- 로그에 민감 데이터 수집
- 로그 인젝션 - 로그에 공격자가 원하는 로그를 삽입하거나 삭제
- 알람 부족

대응 방안

- 로그 무결성 보장 및 중앙 집중화
- 기업내 로그 저장 표준 방식 확립
- 민감 데이터 감지 및 제외
- 주기적 감사

New!

A10: 잘못된 예외 처리 (Mishandling of Exceptional Conditions)

예상치 못하거나 예측 불가능한 상황에 대해 예방, 감지 및 대응하는 데 실패하여 장애, 취약점으로 이어질 때 발생하는 위험입니다.

새로 등장한 이유?

- 복원력(Resilience)에 대한 필요성 증대, 그리고 기존의 분류 체계로 분류하지 못한 에러 관련 문제들을 포함하기 위함
- 커뮤니티에서 중요하게 떠오르는 우려 사항으로 뽑힘

대표적인 취약점

- 에러 정보 노출 - 중요정보, 민감 정보 노출
- Failing Open - 검증 실패시 접근을 허용 - if False: break; if False: break; pass();
- 불충분한 입력 값 처리 - 입력 값을 적절히 처리하지 못하여 예상하지 못한 동작으로 이어지는 경우

대응 방안

- 에러 메세지에 내부 데이터나 민감 데이터가 노출되지 않도록 공통 로직 사용
- 엄격한 입력값 검증
- Fail closed - 검증 성공시 접근을 허용 - if True: pass(); if True: pass2(); fail();

Top 10에 기여할 수 있는 방법

- 데이터를 제공하거나 커뮤니티 설문조사에 참여해주세요.
- 한국어 번역 - 최종 버전이 출시되기 전 Top 10 팀에서 한국어 번역을 위한 기여자들을 모집할 예정입니다. 구체적인 방법은 공개되는대로 공지드리겠습니다.

서울 챕터 카카오톡 커뮤니티

https://open.kakao.com/o/gS5IxXxh

부록 1 - 데이터 순

- 공식: (Max Incidence Rate % * 1000) + (Max Coverage % * 100) + (Avg Exploit * 10) + (Avg Impact * 20) + (Sum Occurrences / 10000) = Risk Score
- CWEs Mapped: The number of CWEs mapped to a category by the Top Ten team.
- Incidence Rate: Incidence rate is the percentage of applications vulnerable to that CWE from the population
- tested by that org for that time period.
- Weighted Exploit: The Exploit sub-score from CVSSv2 and CVSSv3 scores assigned to CVEs mapped to CWEs,
- normalized, and placed on a 10pt scale.
- Weighted Impact: The Impact sub-score from CVSSv2 and CVSSv3 scores assigned to CVEs mapped to CWEs,
- normalized, and placed on a 10pt scale.
- (Testing) Coverage: The percentage of applications tested by all organizations for a given CWE.
- Total Occurrences: Total number of applications found to have the CWEs mapped to a category.
- Total CVEs: Total number of CVEs in the NVD DB that were mapped to the CWEs mapped to a category.

Category	Incidence	Coverage	Exploit	Impact	Occurances	Score	Rank
Software Supply Chain Failures	88.14	65.42	81.7	104.7	21.52	361.42	10
Cryptographic Failures	137.74	100.00	72.3	77.9	166.53	554.56	3
Security Misconfiguration	276.99	100.00	79.6	79.4	71.91	607.89	2
Authentication Failures	158.00	100.00	76.9	88.8	112.07	535.74	5
Software or Data Integrity Failures	89.78	78.52	71.1	95.7	50.13	385.22	9
Memory Management Errors	29.57	55.62	67.5	96.3	22.04	271.08	12
Insecure Design	221.81	88.76	69.6	81.0	72.99	534.19	6
Injection	137.65	100.00	71.5	86.4	140.42	535.96	4
Broken Access Control	201.52	100.00	70.4	76.8	183.97	632.68	1
Logging & Alerting Failures	113.33	85.96	71.9	53.0	26.03	350.20	11
Mishandling of Exceptional Conditio	206.72	100.00	71.1	76.2	76.96	531.00	7
Lack of Application Resilience	200.47	86.01	79.2	69.8	86.51	521.95	8
Weight	1000	100	10	20	10000		

부록 2 - 설문조사 결과

Ranking	Category	Score
#1	Software Supply Chain Failures	522
#2	Software or Data Integrity Failures	273
#3	Logging & Alerting Failures	200
#4	Lack of Application Resilience	193
#5	Mishandling of Exceptional Conditions	178
#6	Memory Management Errors	98