**시큐리티 코딩 개요**

**시큐리티 코딩이란?**

• 시큐리티 코딩은 보안 취약점을 방지하고, 악의적인 공격으로부터 시스템을 보호하기 위한 코딩 기법과 원칙을 의미합니다.

• 이는 보안 중심의 코딩 기법을 통해 소프트웨어 개발 과정에서 발생할 수 있는 보안 위협을 최소화하고, 안전한 코드를 작성하는 것을 목표로 합니다.

• 사전 예방적 접근을 통해 코드가 예상치 못한 상황에서도 안전하게 동작하도록 보장하는 것이 중요합니다.

**중요성 및 필요성**

• 시큐리티 코딩은 민감한 사용자 정보 및 기업 데이터를 보호하고, 데이터 유출 및 손실을 방지하는 데 필수적입니다.

• 보안 취약점으로 인해 시스템이 장애를 겪거나 손상되지 않도록 하여 서비스 가용성과 신뢰성을 확보할 수 있습니다.

• 이를 통해 사용자 및 고객에게 신뢰성 있는 서비스를 제공하고, 기업의 이미지 및 브랜드 가치를 향상시킬 수 있습니다.

**정보 보안 규정 및 표준**

**GDPR (General Data Protection Regulation)**

• 2018년 5월 25일에 발효된 GDPR은 유럽 연합(EU) 내 모든 개인의 데이터 보호 및 프라이버시를 강화하고 통일하기 위해 제정된 법규입니다.

• 데이터 주체의 권리(접근권, 수정권, 삭제권 등), 데이터 처리 원칙, 데이터 보호 책임자의 역할, 데이터 유출 시 통보 의무 등을 포함합니다.

**CCPA (California Consumer Privacy Act)**

• 2018년 6월 28일에 제정되어 2020년 1월 1일부터 시행된 CCPA는 캘리포니아 주에 거주하는 소비자들의 개인정보 보호를 강화하고, 데이터 주체의 권리를 보장하기 위해 제정된 법규입니다.

• 소비자의 접근권, 삭제권, 데이터 판매 거부권, 평등 서비스 및 가격 제공권을 포함하며, 기업은 개인정보 보호정책 공개와 소비자 요청 처리 등의 의무를 가집니다.

**ISMS (Information Security Management System)**

• ISMS는 정보의 기밀성, 무결성, 가용성을 보장하기 위해 정보 보안 관리 체계를 수립하고 유지하는 국제 표준입니다.

• 기업의 정보보호 관리체계에 대한 인증 제도입니다.

• 정보 보안 정책 수립, 위험 평가 및 관리, 보안 통제 조치, 지속적인 모니터링 및 개선을 포함합니다. 인증은 ISO/IEC 27001 표준에 따라 수행됩니다.

**개인정보 보호법 (Personal Information Protection Act, PIPA)**

• 2011년 9월 30일에 제정된 PIPA는 대한민국 내 개인의 개인정보를 보호하고, 개인정보 처리에 대한 권리를 보장하기 위해 제정된 법규입니다.

• 개인정보의 수집 및 이용 동의, 정보 주체의 접근권 및 삭제권, 개인정보 보호 조치 의무, 개인정보 유출 시 통보 의무를 포함합니다.

**시큐리티 코딩의 기본 원칙**

1. **입력 데이터 검증**

• 모든 입력 데이터를 검증하여 악의적인 입력을 방지해야 합니다.

• 이를 위해 화이트리스트 방식을 사용하여 유효한 입력만 허용하고, 정규 표현

식을 통해 입력 형식을 검증합니다.

1. **출력 데이터 인코딩:**

• 출력 데이터를 적절히 인코딩하여 XSS 등의 공격을 방지해야 합니다.

• HTML, JavaScript, SQL 등 각 맥락에 맞는 인코딩을 적용합니다.

1. **암호화**

• 중요한 데이터를 암호화하고 안전하게 저장해야 합니다.

• 전송 중 데이터는 SSL/TLS를 통해 암호화하고, 민감 데이터에 접근 시 로그를 기록합니다**.**

1. **접근 권한 제어**

• 최소 권한 원칙을 준수하여 불필요한 접근을 제한해야 합니다.

• 역할 기반 접근 제어(RBAC)를 적용하고, 세션 관리 및 강력한 인증 메커니즘을 사용합니다.

1. **제한된 정보 제공:**

• 에러 메시지에 민감한 정보를 포함하지 않도록 해야 합니다.

• 일반 사용자에게는 구체적인 에러 정보를 노출하지 않으며, 보안 이벤트 및 접근 시도에 대한 상세한 로깅을 실시합니다.

• 또한, 로그 파일의 무결성을 유지하고 접근을 제어합니다.

**시큐리티 코딩 베스트 프랙티스**

**입력 데이터 검증**

모든 사용자 입력을 신뢰하지 말고, 필터링 및 검증을 통해 악의적인 데이터나 형식을 차단합니다.

**• 화이트리스트 검증:** 허용된 데이터 형식과 값을 정의하고, 입력 데이터가 이 기준에 맞는지 확인합니다.

• **길이 제한:** 입력 필드의 최대 길이를 제한하여 버퍼 오버플로우 공격을 방지합니다.

• **특수 문자 필터링**: SQL 삽입 및 XSS 공격을 방지하기 위해 특수 문자를 필터링하거나 이스케이프합니다.

**XSS (Cross-Site Scripting) 이란?**

XSS는 공격자가 악의적인 스크립트를 웹 페이지에 삽입하여 사용자의 브라우저에서. 실행되도록 하는 사이버 공격입니다. 주로 자바스크립트를 사용하며, 공격자가 사용자 정보를 탈취하거나 악성 코드를 배포하는 데 사용됩니다.

**데이터 인코딩**

웹 애플리케이션에서 사용자에게 표시되는 데이터를 안전하게 변환하여, 악의적인 코드가 실행되지 않도록 합니다.

**• HTML 인코딩:** HTML 콘텐츠에 특수 문자를 HTML 엔티티로 변환하여 XSS 공격을 방지합니다.

**• URL 인코딩:** URL에 포함되는 데이터는 URL 인코딩하여 쿼리 문자열 및 경로를 안전하게 처리합니다.

• **문자열 인코딩:** 데이터베이스에서 반환되는 문자열을 적절하게 인코딩하여 SQL 인젝션 공격을 방지합니다.

**SQL 인젝션 (SQL Injection) 이란?**

SQL 인젝션은 공격자가 악의적인 SQL 코드를 애플리케이션의 입력 필드에 삽입하여 데이터베이스의 쿼리를 조작하는 공격입니다. 이로 인해 데이터베이스의 정보를 조회하거나 수정, 삭제할 수 있으며, 때로는 데이터베이스 서버에 대한 권한을 얻을 수 있습니다.

**데이터 보호**

데이터의 기밀성, 무결성 및 가용성을 유지하기 위한 조치를 취합니다.

• **암호화:** 저장 데이터와 전송 데이터를 암호화하여 무단 접근과 데이터 유출을 방지합니다.

• **접근 제어:** 데이터에 대한 접근 권한을 관리하고, 필요한 권한만 부여합니다.

**• 백업:** 정기적인 데이터 백업을 통해 데이터 손실에 대비합니다.

**암호화 및 해싱**

암호화와 해싱은 데이터를 안전하게 보호하기 위한 두 가지 주요 기법입니다. 암호화는 데이터를 변형하여 기밀성을 유지하고, 해싱은 데이터의 무결성을 확인하는 데 사용됩니다.

**암호화**

**• 대칭 암호화:** 동일한 키를 사용하여 데이터를 암호화하고 복호화합니다. 예시) AES, DES

• **비대칭 암호화:** 공개키와 비공개키를 사용하여 데이터를 암호화하고 복호화합니다. 예시) RSA, ECC

**해싱**

**• 해시 알고리즘:** 데이터의 무결성을 보장하기 위해 SHA-256, SHA-3 등의 강력한 해시 알고리즘을 사용합니다.

**• 솔팅:** 해시 값에 임의의 데이터를 추가하여 동일한 입력값에 대해 다른 해시 값을 생성합니다.

**인증 및 권한 부여**

사용자 및 시스템의 신원을 확인하고, 권한에 따라 자원에 대한 접근을 제어합니다.

• **강력한 인증:** 다단계 인증(MFA) 및 강력한 암호 정책을 사용하여 사용자 인증을 강화합니다.

• **권한 최소화:** 사용자가 필요로 하는 최소한의 권한만 부여하여 권한 남용을 방지합니다.

• **세션 관리:** 세션 타임아웃 및 재인증 절차를 구현하여 세션 하이재킹을 방지합니다.

**에러 및 로깅 관리**

시스템 오류와 이벤트를 기록하고, 보안 위협에 대한 분석과 문제 해결을 지원합니다.

• **에러 메시지:** 상세한 오류 메시지와 시스템 내부 정보를 사용자에게 노출하지 않도록 합니다.

• **로깅:** 중요한 보안 이벤트와 시스템 활동을 기록하여 사건 발생 시 분석과 대응을 용이하게 합니다.

• **로그 보호:** 로그 파일에 대한 접근 제어를 강화하고, 로그 데이터의 무결성을 보호합니다.

• **모니터링:** 로그를 실시간으로 모니터링하여 이상 징후를 조기에 탐지하고 대응합니다.

**DevSecOps 및 지속적인 보안 통합**

**DevSecOps의 개념과 중요성**

• DevSecOps는 개발(Development), 보안(Security), 운영 (Operations)의 통합을 통해 소프트웨어 개발 주기 전반에 걸쳐 보안을 강화하는 접근 방식입니다.

• **중요성**

• 개발 초기 단계에서 보안을 통합하여 취약점을 조기에 발견하고 수정할 수 있음

• 보안 검토와 테스트가 자동화되어 배포 속도를 저하시키지 않으면서도 안전성을 보장

• 개발, 보안, 운영 팀 간의 원활한 협업을 통해 보안 문제를 신속히 해결할 수 있음

• 보안 업데이트와 패치를 자동화하여 지속적으로 시스템을 보호할 수 있음

**-> CI/CD 파이프라인에 소프트웨어 보안 검증 프로세스를 추가하여 소프트웨어 개발 주기 전반에 걸쳐 보안을 강화한다.**

**CI/CD 파이프라인에 보안 통합하기**

• CI/CD(지속적 통합 및 지속적 배포) 파이프라인에 보안을 통합하여 코드의 빌드, 테스트, 배포 과정에서 보안을 검토하고 강화하는 과정입니다.

• **방법**

• **정적 분석 도구 통합:** CI 파이프라인에 정적 분석 도구를 통합하여 코드가 커밋될 때마다 보안 취약점을 자동으로 분석

• **동적 분석 도구 통합:** CD 파이프라인에 동적 분석 도구를 통합하여 배포된 애플리케이션의 런타임 보안 테스트를 수행

• **보안 정책 적용:** CI/CD 설정에 보안 정책을 적용하여 모든 빌드와 배포가 보안 기준을 만족하도록 보장

• **비밀 관리:** CI/CD 파이프라인에서 민감한 정보를 안전하게 관리하고 전달하기 위한 비밀 관리 도구 사용

**SBOM (Software Bill of Materials)**

• SBOM은 소프트웨어의 모든 구성 요소, 라이브러리, 의존성 및 해당 메타데이터를 기록한 목록입니다.

• 이를 통해 소프트웨어의 구성 요소와 그 출처를 명확히 하고 보안 및 컴플라이언스 관리를 용이하게 합니다.

• **중요성**

• 사용 중인 라이브러리 및 의존성의 취약점을 빠르게 식별하고 대응할 수 있음

• 소프트웨어의 구성 요소가 사용하는 라이센스 규정을 준수하는지 검토할 수 있음

• 보안 사고 발생 시 소프트웨어 구성 요소를 분석하여 문제의 범위를 파악하고 대응할 수 있음

• 소프트웨어 공급망의 투명성을 확보하여 신뢰성을 높일 수 있음

**CycloneDX와 SPDX**

• **CycloneDX:** SBOM을 생성하고 관리하기 위한 오픈 소스 표준

• **SPDX:** 소프트웨어 라이센스 및 컴포넌트 정보를 문서화하는 표준

• 다양한 SBOM을 사용하여 소프트웨어 구성 요소에 대한 보고서를 작성하고, 보안 및 라이센스 관련 문제를 명확히 전달할 수 있습니다.

• CI/CD 파이프라인에 SBOM 생성 및 검토 과정을 통합하여 이러한 내용을 자동화할 수 있습니다.

**CycloneDX**

CycloneDX는 OWASP에서 개발한 오픈 소스 SBOM 형식으로, 소프트웨어 공급망의 위험 식별과 취약점 관리를 목적으로 합니다.

• **특징**

• 정보 포맷: XML 및 JSON 포맷을 지원

• 자동화 지원: 소프트웨어 빌드 주기 전반에 걸쳐 SBOM 생성 자동화를 지원합니다.

• 취약점 식별: 구성 요소의 보안 취약점을 식별하는 데 중점을 둡니다.

• **주요 필드**

• BOM 메타데이터: 소프트웨어 제조업체/공급업체 및 도구의 메타데이터

• 구성 요소: 소프트웨어의 모든 구성 요소에 대한 설명

• 서비스: 외부 API, 엔드포인트 URL 및 인증 요구 사항

• 종속성: 소프트웨어 구성 요소 간의 직접적이고 전이적인 관계

• 확장: 향후 기능 확장을 위한 데이터 제공

• 사용 예시: 취약점 관리 및 보안 모니터링, 자동화된 SBOM 생성

**SPDX**

SPDX는 Linux Foundation이 개발한 오픈 소스 프로젝트로, 주로 소프트웨어 라이센스 관리와 규정 준수를 위해 설계되었습니다.

**• 특징**

• 정보 포맷: JSON, RDF, Tag/Value 포맷을 지원

• 라이센스 준수: 소프트웨어의 라이센스 정보를 명확히 기록하여 법적 요구사항을 준수합니다.

• 상호 운용성: 다른 SBOM 형식과의 호환성을 높이기 위한 기능을 포함합니다.

• **주요 필드**

• 문서 생성 정보: SBOM 문서의 생성일 및 처리 도구 정보

• 패키지 정보: 소프트웨어 패키지와 관련된 정보

• 파일 정보: 패키지에 포함된 파일의 이름, 라이센스, 저작권 정보

• 스니펫 정보: 코드 조각이 다른 소스에서 가져온 경우

• 관계 및 주석: 문서 내 구성 요소 간의 관계를 설명합니다.

• 사용 예시: 라이센스 준수 확인 및 소프트웨어 구성 요소에 대한 세부 정보 문서화