**프로젝트명**

개발표준정의서

(Backend 개발 가이드)

[산출물관리번호]

Ver. 0.70

2022.01.27

개 정 이 력

| **버전** | **작성일** | **변경 내용** | **작성자** | **승인자** |
| --- | --- | --- | --- | --- |
| 0.70 | 2022.01.27 | 최초 작성 | 이상기 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Contents

[Table of Contents 3](#_Toc94196829)

[1. 문서 개요 5](#_Toc94196830)

[1.1. 개요 5](#_Toc94196831)

[1.1.1. 목적 5](#_Toc94196832)

[1.1.2. 대상 5](#_Toc94196833)

[1.1.3. 내용 5](#_Toc94196834)

[1. 프로젝트 구조 6](#_Toc94196835)

[1.1. 프로젝트 6](#_Toc94196836)

[1.2. 패키지 6](#_Toc94196837)

[2. API 개발 7](#_Toc94196838)

[2.1. REST API 7](#_Toc94196839)

[2.2. SWAGGER 7](#_Toc94196840)

[3. 서비스 개발 8](#_Toc94196841)

[3.1. Transaction 8](#_Toc94196842)

[3.1.1. 적용 기준 8](#_Toc94196843)

[3.1.2. Transaction 분리. 8](#_Toc94196844)

[3.1.3. Transaction 분리 적용예 8](#_Toc94196845)

[3.2. Big Decimal 9](#_Toc94196846)

[4. Java Code Convention 11](#_Toc94196847)

[5. 코드 인스펙션 12](#_Toc94196848)

[5.1. SonarLint Rules 12](#_Toc94196849)

[6. SW 보안 취약성 점검 진단 가이드 13](#_Toc94196850)

[6.1. 설계단계 보안항목 13](#_Toc94196851)

[6.1.1. 입력데이터 검증 및 표현 13](#_Toc94196852)

[6.1.2. 보안기능 13](#_Toc94196853)

[6.1.3. 에러처리 14](#_Toc94196854)

[6.1.4. 세션통제 14](#_Toc94196855)

[6.2. 구현 단계 SW보안 취약성 점검 항목 14](#_Toc94196856)

[6.2.1. 입력데이터 검증 및 표현 14](#_Toc94196857)

[6.2.2. 보안기능 15](#_Toc94196858)

[6.2.3. 시간 및 상태 16](#_Toc94196859)

[6.2.4. 에러처리 16](#_Toc94196860)

[6.2.5. 코드오류 16](#_Toc94196861)

[6.2.6. 캡슐화 17](#_Toc94196862)

[6.2.7. API 오용 17](#_Toc94196863)

[6.3. 개인정보의 안정성 확보조치 17](#_Toc94196864)

[6.3.1. 개요 17](#_Toc94196865)

[6.3.2. 민감성 정보를 고려한 개발표준 18](#_Toc94196866)

# 문서 개요

## 개요

### 목적

본 문서는 {프로젝트명}(이하 ‘프로젝트’라 한다.)의 성공적인 구축을 위하여 {프로젝트명} 표준 프레임워크 기반의 소프트웨어 개발 과정에 참여하는 모든 Backend 프로그램 개발자에게 필요한 개발 표준을 전달할 수 있도록 합니다. 이를통해 Backend 프로그램 개발자의 개발 효율을 향상시키고 표준화되고 패턴화된 산출물(프로그램 소스)를 통해 향후 운영 효율을 향상시키는 것에 목적이 있습니다

### 대상

본 문서는 아래와 같은 관련자를 대상으로 한다.

- 업무 개발자
- 공통 모듈 개발자

- 프레임워크 관리자

### 내용

본 문서는 아래와 같은 내용을 설명한다.

- API 개발

- 서비스 개발

- Java Code Style

- Java Code Inspection

- SW 보안 취약점 진단 가이드

1. 프로젝트 구조

프로젝트 구조는 다음과 같다.

* 1. 프로젝트

- 구매 시스템

- 협력사 포탈

* 1. 패키지

{프로젝트명}의 표준 Package는 ‘xxx.xxx.xxxx’ 패키지를 시작으로 다음과 같은 구조로 구성한다.

L2 : 업무 코드 사용.

L3 : Layer 구분. ( controller, service, dao, model )

L4 : Controller의 경우 추후 확장성을 고려하여 Client 구분자를 위치 시킴.( 기본:web, RestAPI와 같이 웹서비스를 통해 인터페이스 하는 외부시스템코드, PDA와 같은 Device 코드 )

|  |  |  |
| --- | --- | --- |
| PackageLevel | Package Name | 비고 |
| 1 | *common* | *L2 시스템* |
| 2 | *controller* | *Layer 구분* |
| 3 | *web* | *Client 구분* |
| 2 | service | *Layer 구분* |
| 2 | dao | *Layer 구분* |
| 2 | model | *Layer 구분* |

1. API 개발

* 1. REST API

추후 상세하게 정의

* 1. SWAGGER

추후 상세하게 정의

1. 서비스 개발
	1. Transaction
		1. 적용 기준

- Transaction은 서비스 단위로 적용시킨다.

- Transaction 보장을 위해 Database 변경 기능이 있는 서비스에 대해서 @Transactional annotation을 사용한다.

* + 1. Transaction 분리.

- 트랜잭션 분리는 서비스의 정상종료(Commit)나 예외종료(Rollback)와 상관없이 서비스 내의 특정 데이터 처리를 별도의 트랜잭션으로 처리할 때 사용하는 방법이다. 트랜잭션 분리된 메소드 내의 갱신데이터는 해당 메소드가 종료되는 시점에 Commit된다

- @Transactional 사용시 propagation 속성을 Propagation.REQUIRES\_NEW 로 설정하여 적용시킨다.

* + 1. Transaction 분리 적용예

- 비밀번호가 틀릴 때, 서비스는 예외종료처리하지만, 비밀번호오류횟수를 +1 할 때,

@Transactiona (propagation=Propagation.REQUIRES\_NEW) //트랜잭션분리

 private String \_createNewSpvsrAprvl(AprvlCndCnfrmtnIn in, IOmmObject srvcOmm,

 List<String> matchedCnds) throws BizApplicationException,

 AprvlApplicationException {

 AprvlInfoIn aprvlInfoIn = new AprvlInfoIn();

 aprvlInfoIn.setInstCd(cmnUtilMngr.getInstCd());// set [기관코드]

 aprvlInfoIn.setAprvlTrgtSrvcCd(cmnUtilMngr.getSrvcCd());// set [승인대상서비스코드]

 aprvlInfoIn.setWflowInstncId(null);// set [워크플로우인스턴스식별자]

 aprvlInfoIn.setAprvlTrgtBizIdCntnt(in.getAprvlTrgtBizIdCntnt());// set [승인대상업무식별키내용]

 aprvlInfoIn.setAprvlDemandStaffId(cmnUtilMngr.getStaffId());// set [승인요청스태프식별자]

 aprvlInfoIn.setAprvlStsCd(CCM01.WFLOW\_APRVL\_STS\_CD\_BEF\_WFLOW);// set [승인상태코드] 09.workflow 요청 전

 aprvlInfoIn.setAprvlAplctnDt(cmnUtilMngr.getTxDt());// set [승인신청년월일]

 aprvlInfoIn.setAprvlAplctnCntnt(srvcOmm.toString());// set [승인신청내용]

 aprvlInfoIn.setAprvlTpCd(CCM01.WFLOW\_TP\_CD\_APRVL);// set [승인유형코드] 11905 Workflow 유형코드 01 결재프로세스

 aprvlInfoIn.setLastAprvlDt(null);// set [최종승인년월일]

 aprvlInfoIn.setLastChngGuid(null);// set [최종변경GUID]

 /\* 결재VO 호출 \*/

 String aprvlId = aprvlMngr.createAprv(aprvlInfoIn);

 /\* 결재발생조건 DBIO \*/

 for ( String matchedcnd : matchedCnds ) {

 CmAprvlIssueCndDIO indao = new CmAprvlIssueCndDIO();

 indao.setAprvlId(aprvlId);// set [승인식별자]

 indao.setAprvlCndNbr(matchedcnd);// set [결제조건번호]

 cmnUtilMngr.setHdrClmn(indao); // Institution Code, National Code, GUID, Last Change TimeStamp

 cmAprvlIssueCndD.insert(indao); //결재발생조건DBIO

 }

 return aprvlId;

 }

* 1. Big Decimal
* 금액 연산은 BigDecimal 을 사용하는 것을 권장한다.
* 초기화는 String 타입으로 하고, double이나 int 형으로 사용하지 않는다.
* BigDecimal 사용하지 않은 경우의 문제점은 다음과 같다.

- 정확한 소수점 연산이 되지 않은 경우가 있으므로(아래 소스 참고), 부동소수점 유형을 사용하는 경우, equal연산이 의도한 바로 적용되지 않을 수 있음(ex. 단말 입력 값 0.5 != DB 조회 값 0.5)

- BigDecimal을 사용하지 않은 경우 나누기 연산 등을 마지막에 수행하는 등, 연산 순위를 고려해야 한다.

import java.math.BigDecimal;

public class Exam1 {

 /\*\*

 \* @param args

 \*/

 public static void main(String[] args) {

 // Double 연산의 문제

 BigDecimal s1 = new BigDecimal("2.0");

 BigDecimal s2 = new BigDecimal("1.1");

 System.out.println(s1.subtract(s2)); // 결과 : 0.9

 System.out.println(2.0 - 1.1); // 결과 : 0.8999999999999999

 // BigDecimal의 초기화는 문자열로 해야됨.

 //그렇지 않고 double 하게 되면 위의 문제가 그대로 발생.

 BigDecimal x = new BigDecimal(2.0 - 1.1); // 잠재 오류 있음

 System.out.println(x);

 // 결과 : 0.899999999999999911182158029987476766109466552734375

 // 무한소수 처리

 BigDecimal a = new BigDecimal("3");

 BigDecimal b = new BigDecimal("1");

 // System.out.println(b.divide(a));

// Exception 발생 Non-terminating decimal expansion; no exact representable decimal result.

 System.out.println(b.divide(a, 3, BigDecimal.ROUND\_DOWN));

 // 결과 : 0.333

 }

}

1. Java Code Convention

Java Coding Convention은 Java를 이용하여 코딩을 하는 프로그래머 사이의 규칙이다.

소프트웨어를 개발하는 일련의 모든 과정에 들어가는 비용 중 80%가 유지보수에 사용되며, 소프트웨어를 직접 개발한 개발자가 그 소프트웨어의 유지보수를 담당하는 경우는 거의 보기 힘들다.

코딩 규칙을 지키면 다른 개발자가 그 소스 코드를 처음 보았을 때, 더 빠른 시간 안에 완벽하게 이해할 수 있도록 도와주기 때문에, 소프트웨어의 가독성이 높아진다.

*‘Java Coding Convention’ 관련되어 상세 내용은 개발표준정의서(Online) 문서와 함께 배포한 [개발표준정의서(JavaCode\_Conventions)\_V1.10(sample).docx] 문서를 참고한다.*

1. 코드 인스펙션

개발자가 작성한 소스는 {발주사}에서 정의하는 코드 인스펙션 프로세스로 검증되어야 한다.

* 1. SonarLint Rules

소스 코드 인스펙션 전문도구로써 보안 코딩, 버그 위험성 관점에서 검증한다.

추후 상세 기술.

1. SW 보안 취약성 점검 진단 가이드

'SW개발보안'은 SW개발과정에서 개발자의 실수, 논리적 오류 등으로 인해 발생될 수 있는 보안 취약점, 보안약점들을 최소화하여 사이버 보안 위협에 대응할 수 있는 안전한 SW를 개발하기 위한 일련의 보안 활동을 의미한다. 즉, SW개발 생명주기(SDLC, Software Development Life Cycle)의 각 단계별로 요구되는 보안활동을 수행함으로써 안전한 소프트웨어를 만들 수 있도록 한다.

설계단계에서 개발 보안을 적용할 때 얻을 수 있는 효과는 아래와 같다.

* 개발보안을 일관성 있게 적용
* 구현단계에서 개발보안 항목을 명확히 알 수 있음

*‘SW개발보안’ 관련되어 상세 내용 및 개발 시 참고해야 할 샘플은 개발표준정의서(Online) 문서와 함께 배포한 [한국인터넷진흥원\_소프트웨어\_개발보안\_가이드(2019.11)\_V1.00.pdf] 문서를 참고한다.*

## 설계단계 보안항목

### 입력데이터 검증 및 표현

### 보안기능

인증, 접근통제, 권한관리, 비밀번호 등의 정책이 적절하게 반영될 수 있도록 설계한다.

|  |  |  |
| --- | --- | --- |
| 보안약점 항목 | 설명 | 설계반영 업무 |
| SR2-1. 인증 대상 및 방식 | 중요정보·기능과 인증방식을 정의하고, 정의된 중요정보 접근 및 중요기능 수행 허용을 위해 인증 기능이 우회되지 않고 수행될 수 있도록 설계 | F/W, 업무총괄 |
| SR2-2. 인증 수행 제한 | 인증 반복시도 제한 및 인증실패 등에 대한 인증제한 기능 설계 | 업무총괄 |
| SR2‐3 비밀번호 관리 | 안전한 비밀번호 조합규칙(비밀번호 길이, 허용문자 조합 등)을 설정하고, 안전한 저장 정책, 재설정 및 변경 정책, 패스워드 관리규칙(주기적 변경 등)이 적용되도록 설계 | 업무총괄 |
| SR2‐4. 중요자원 접근통제 | 중요자원(프로그램 설정, 민감한 사용자 데이터 등)을 정의하고, 정의된 중요자원에 대한 접근을 통제 하는 신뢰할 수 있는 방법(권한관리 포함) 및 접근통제 실패시 대응방안 설계 | 업무총괄 |
| SR2‐5. 암호키 관리 | 암호키 생성, 분배, 접근, 파기 등 안전하게 암호키 생명주기를 관리할 수 있는 방법 설계 | DB암호화솔루션 |
| SR2‐6. 암호연산 | 국제표준 또는 검증필 프로토콜로 등재된 안전한 암호 알고리즘을 선정하여 충분한 암호키 길이, 솔트, 충분한 난수값을 기반으로 암호연산 수행방법 설계 | DB암호화솔루션 |
| SR2‐7. 중요정보 저장 | 중요정보(비밀번호, 개인정보 등) 저장시 안전한 저장 및 관리방법 설계 | 업무총괄 |
| SR2‐8. 중요정보 전송 | 중요정보(비밀번호, 개인정보 등) 전송시 안전한 전송방법 설계 | 업무총괄 |

### 에러처리

에러 또는 오류상황을 처리하지 않거나 불충분하게 처리되어 중요정보 유출 등 보안약점이 발생하지 않도록 설계한다.

|  |  |  |
| --- | --- | --- |
| 보안약점 항목 | 설명 | 설계반영 업무 |
| SR3-1. 예외처리 | 오류메시지에 중요정보(개인정보, 시스템 정보, 민감 정보 등)가 포함되어 출력되거나, 에러 및 오류가 부적절하게 처리되어 의도치 않은 상황이 발생하는 것을 막기 위한 안전한 방안 설계 | F/W |

### 세션통제

세션을 사용할 경우에 해당.

HTTP를 이용하여 연결을 유지하는 경우 세션을 안전하게 할당하고 관리하여 세션정보노출이나 세션 하이재킹과 같은 침해사고가 발생하지 않도록 설계한다.

|  |  |  |
| --- | --- | --- |
| 보안약점 항목 | 설명 | 설계반영 업무 |
| SR4-1. 세션통제 | 다른 세션간 데이터 공유금지, 세션 ID 노출금지, (재)로그인시 세션ID 변경, 세션종료(비활성화, 유효기간 등) 처리 등 세션을 안전하게 관리할 수 있는 방안 설계 | F/W |

## 구현 단계 SW보안 취약성 점검 항목

### 입력데이터 검증 및 표현

|  |  |  |
| --- | --- | --- |
| No. | 보안약점 | 설명 |
| 1 | SQL 삽입  | 검증되지 않은 외부입력값이 SQL 쿼리문 생성에 사용되어 악의적인 쿼리가 실행될 수 있는 보안약점 |
| 2 | 경로 조작 및 자원 삽입  | 검증되지 않은 외부입력값이 시스템 자원 접근경로 또는 자원제어에 사용되어 공격자가 입력값을 조작해 공격할 수 있는 보안약점 |
| 3 | 크로스사이트 스크립트  | 검증되지 않은 외부입력값에 의해 사용자 브라우저에서 악의적인 스크립트가 실행될 수 있는 보안약점 |
| 4 | 운영체제 명령어 삽입  | 검증되지 않은 외부입력값이 운영체제 명령문 생성에 사용되어 악의적인 명령어가 실행될 수 있는 보안약점 |
| 5 | 위험한 형식 파일 업로드  | 파일의 확장자 등 파일형식에 대한 검증 없이 업로드를 허용하여 발생할 수 있는 보안약점 |
| 6 | 신뢰되지 않는 URL 주소로 자동접속 연결 | 검증되지 않은 외부입력값이 URL 링크 생성에 사용되어 악의적인 사이트로 자동 접속될 수 있는 보안약점 |
| 7 | XQuery 삽입  | 검증되지 않은 외부입력값이 XQuery 쿼리문 생성에 사용되어 악의적인 쿼리가 실행될 수 있는 보안약점 |
| 8 | XPath 삽입  | 검증되지 않은 외부입력값이 XPath 쿼리문 생성에 사용 되어 악의적인 쿼리가 실행될 수 있는 보안약점 |
| 9 | LDAP 삽입  | 검증되지 않은 입력값이 LDAP 명령문 생성에 사용되어 악의적인 명령어가 실행될 수 있는 보안약점 |
| 10 | 크로스사이트 요청 위조 | 검증되지 않은 외부입력값에 의해 브라우저에서 악의적인 스크립트가 실행되어 공격자가 원하는 요청(Request)이 다른 사용자(관리자 등)의 권한으로 서버로 전송되는 보안약점 |
| 11 | HTTP 응답분할  | 검증되지 않은 외부입력값이 HTTP 응답헤더에 삽입되어 악의적인 코드가 실행될 수 있는 보안약점 |
| 12 | 정수형 오버플로우  | 정수를 사용한 연산의 결과가 정수값의 범위를 넘어서는 경우, 프로그램이 예기치 않게 동작될 수 있는 보안약점 |
| 13 | 보안기능 결정에 사용되는 부적절한 입력값 | 검증되지 않은 입력값이 보안결정(인증, 인가, 권한부여 등)에 사용되어 보안 메커니즘 우회 등을 야기할 수 있는 보안약점 |
| 14 | 메모리 버퍼 오버플로우  | 메모리 버퍼의 경계값을 넘어서 메모리값을 읽거나 저장하여 예기치 않은 결과를 발생시킬 수 있는 보안약점 |
| 15 | 포맷 스트링 삽입  | printf 등 외부입력값으로 포맷스트링을 제어할 수 있는 함수를 사용하여 발생 할 수 있는 보안약점 |

### 보안기능

|  |  |  |
| --- | --- | --- |
| No. | 보안약점 | 설명 |
| 1 | 적절한 인증 없는 중요 기능 허용 | 적절한 인증 없이 중요정보(금융정보, 개인정보, 인증정보 등)를 열람(또는 변경)할 수 있게 하는 보안약점 |
| 2 | 부적절한 인가  | 적절한 접근제어 없이 외부입력값을 포함한 문자열로 중요자원에 접근할 수 있는 보안약점 |
| 3 | 중요한 자원에 대한 잘못된 권한 설정 | 중요자원(프로그램 설정, 민감한 사용자 데이터 등)에 대한 적절한 접근권한을 부여하지 않아, 인가되지 않은 사용자 등에 의해 중요정보가 노출·수정되는 보안약점 |
| 4 | 취약한 암호화 알고리즘 사용 | 중요정보(금융정보, 개인정보, 인증정보 등)의 기밀성을 보장할 수 없는 취약한 암호화 알고리즘을 사용하여 정보가 노출될 수 있는 보안약점 |
| 5 | 중요정보 평문저장  | 중요정보(비밀번호, 개인정보 등)를 암호화하여 저장하지 않아 정보가 노출될 수 있는 보안약점 |
| 6 | 중요정보 평문전송 | 중요정보(비밀번호, 개인정보 등) 전송시 암호화하지 않거나 안전한 통신채널을 이용하지 않아 정보가 노출될 수 있는 보안약점 |
| 7 | 하드코드된 비밀번호  | 소스코드 내에 비밀번호가 하드코딩 되어 소스코드 유출시 노출 우려 및 주기적 변경 등 수정(관리자 변경 등)이 용이하지 않는 보안약점 |
| 8 | 충분하지 않은 키 길이 사용 | 데이터의 기밀성, 무결성 보장을 위해 사용되는 키의 길이가 충분하지 않아 기밀정보 누출, 무결성이 깨지는 보안약점 |
| 9 | 적절하지 않은 난수 값 사용 | 예측 가능한 난수사용으로 공격자로 하여금 다음 숫자 등을 예상하여 시스템 공격이 가능한 보안약점 |
| 10 | 하드코드된 암호화 키  | 소스코드 변경이 용이하지 내에 암호화키가 않는 보안약점 하드코딩 되어 소스코드 유출시 노출 우려 및 키 변경이 용이하지 않는 보안약점 |
| 11 | 취약한 비밀번호 허용  | 비밀번호 조합규칙(영문, 숫자, 특수문자 등) 미흡 및 길이가 충분하지 않아 노출될 수 있는 보안약점 |
| 12 | 사용자 하드디스크에 저장되는 쿠키를 통한 정보노출 | 쿠키(세션 ID, 사용자 권한정보 등 중요정보)를 사용자 하드디스크에 저장함으로써 개인정보 등 기밀정보가 노출될 수 있는 보안약점 |
| 13 | 주석문 안에 포함된 시스템 주요정보 | 소스코드내의 주석문에 인증정보 등 시스템 주요정보가 포함되어 소스코드 유출시 노출될 수 있는 보안약점 |
| 14 | 솔트 없이 일방향 해시 함수 사용 | 공격자가 솔트 없이 생성된 해시값을 얻게 된 경우, 미리 계산된 레인보우 테이블을 이용하여 원문을 찾을 수 있는 보안약점 |
| 15 | 무결성 검사 없는 코드 다운로드 | 원격으로부터 소스코드 또는 실행파일을 무결성 검사 없이 다운로드 받고 이를 실행하는 경우, 공격자가 악의적인 코드를 실행할 수 있는 보안약점 |
| 16 | 반복된 인증시도 제한 기능 부재 | 인증시도의 수를 제한하지 않아 공격자가 무작위 인증시도를 통해 계정접근 권한을 얻을 수 있는 보안약점 |

### 시간 및 상태

|  |  |  |
| --- | --- | --- |
| No. | 보안약점 | 설명 |
| 1 | 경쟁조건 : 검사 시점과 사용 시점(TOCTOU) | 멀티 프로세스 상에서 자원을 검사하는 시점과 사용하는 시점이 달라서 발생하는 보안약점 |
| 2 | 종료되지 않는 반복문 또는 재귀함수 | 종료조건 없는 제어문 사용으로 반복문 또는 재귀함수가 무한히 반복되어 발생할 수 있는 보안약점 |

### 에러처리

|  |  |  |
| --- | --- | --- |
| No. | 보안약점 | 설명 |
| 1 | 오류 메시지를 통한 정보 노출 | 개발자가 생성한 오류 메시지에 시스템 내부구조 등이 포함되어 민감한 정보가 노출될 수 있는 보안약점 |
| 2 | 오류 상황 대응 부재  | 시스템에서 발생하는 오류 상황을 처리하지 않아 프로그램 실행정지 등 의도하지 않은 상황이 발생할 수 있는 보안약점 |
| 3 | 부적절한 예외 처리  | 예외에 대한 부적절한 처리로 인해 의도하지 않은 상황이 발생될 수 있는 보안 약점 |

### 코드오류

|  |  |  |
| --- | --- | --- |
| No. | 보안약점 | 설명 |
| 1 | Null Pointer 역참조  | Null로 설정된 변수의 주소값을 참조했을 때 발생하는 보안약점 |
| 2 | 부적절한 자원 해제  | 사용된 자원을 적절히 해제하지 않으면 자원 누수 등이 발생하고, 자원이 부족하여 새로운 입력을 처리할 수 없게 되는 보안약점 |
| 3 | 해제된 자원 사용  | 메모리 등 해제된 자원을 참조하여 예기치 않은 오류가 발생될 수 있는 보안약점 |
| 4 | 초기화되지 않은 변수 사용  | 변수를 초기화하지 않고 사용하여 예기치 않은 오류가 발생될 수 있는 보안약점 |

### 캡슐화

|  |  |  |
| --- | --- | --- |
| No. | 보안약점 | 설명 |
| 1 | 잘못된 세션에 의한 데이터 정보노출 | 잘못된 세션에 의해 인가되지 않은 사용자에게 중요정보가 노출될 수 있는 보안약점 |
| 2 | 제거되지 않고 남은 디버그 코드 | 디버깅을 위해 작성된 코드를 통해 인가되지 않은 사용자에게 중요정보가 노출될 수 있는 보안약점 |
| 3 | 시스템 데이터 정보노출 | 사용자가 볼 수 있는 오류 메시지나 스택 정보에 시스템 내부 데이터나 디버깅 관련 정보가 공개되는 보안약점 |
| 4 | Public 메서드로부터 반환된 Private 배열 | Private로 선언된 배열을 Public으로 선언된 메서드를 통해 반환(return)하면, 그 배열의 레퍼런스가 외부에 공개되어 외부에서 배열이 수정될 수 있 는 보안약점 |
| 5 | Private 배열에 Public 데이터 할당 | Public으로 선언된 데이터 또는 메서드의 파라미터가 Private로 선언된 배 열에 저장되면, Private 배열을 외부에서 접근할 수 있게 되는 보안약점 |

### API 오용

|  |  |  |
| --- | --- | --- |
| No. | 보안약점 | 설명 |
| 1 | DNS lookup에 의존한 보안결정 | DNS는 공격자에 의해 DNS 스푸핑 공격 등이 가능하므로 보안 결정을 DNS 이름에 의존할 경우, 보안결정 등이 노출되는 보안약점 |
| 2 | 취약한 API 사용  | 취약하다고 알려진 함수를 사용함으로써 예기치 않은 보안위협에 노출될 수 있는 보안약점 |

## 개인정보의 안정성 확보조치

### 개요

{프로젝트명} 구축시, ‘개인정보보호법’과 ‘개인정보의 안정성 확보조치 기준’에 따라 중요정보는 암호화하여 별도 관리한다.

* 중요정보 유형 및 관리 방식

|  |  |  |  |
| --- | --- | --- | --- |
| 유형 | 관리대상 | 암호화 방식 | 비고 |
| 비밀번호 | 비밀번호 | SHA-256 방식일방향 암호화 수행 | Web token에 암호화 하여 사용. ( Web token 사용 여부 추후 확정 필요 ) |
| 개인식별정보 | 주민번호계좌번호법인카드아이디 | SHA-256방식양방향 암호화 수행 | Meta 표준단어를 바탕으로 Framework에서 자동 암/복호화를 수행함. 개발자 별도 조치 필요 없음 |

모든 중요정보의 접근에 대해서는 별도 접근기록 보관을 해야 한다

### 민감성 정보를 고려한 개발표준

표준 단어 중 민감성 정보와 관련된 단어는 별도로 관리하여 Framework에서 공통 처리한다.

암호화, 마스킹 처리는 개인정보 보호법에 따른다.