[IT Essential] OWASP top 10과 시큐어 코딩

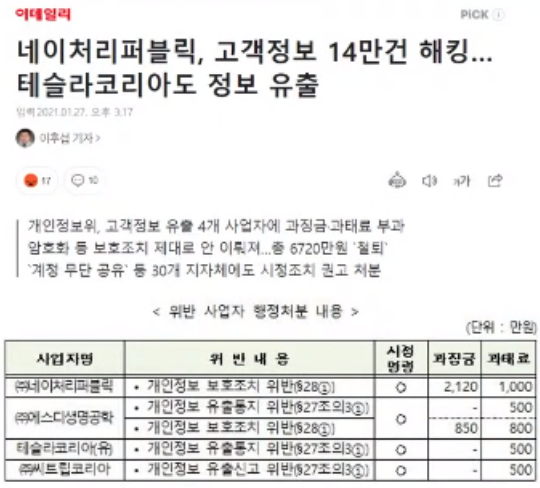
> 진행자: 박종철 컨설턴트님
>
> 날짜: 2021-02-17

목차

1. 학습 목표

1. 시큐어 코딩과 OWASP TOP 10의 의미를 알 수 있고 코드에 적용 할 수 있다.
2. 해시 함수(Hash Function)에 대해서 알 수 있다.
3. 실무에서 쓰이는 유용한 팁과 보안 위배 코딩에 대해서 알 수 있다.

2. 시큐어 코딩과 OWASP TOP 10



이데일리 PICK

네이처리퍼블릭, 고객정보 14만건 해킹… 테슬라코리아도 정보 유출

입력 2021.01.27. 오후 3:17

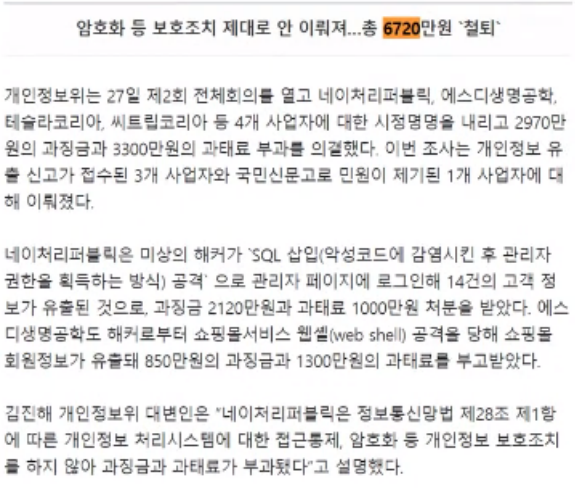
이후섭 기자

개인정보위, 고객정보 유출 4개 사업자에 과징금·과태료 부과
암호화 등 보호조치 제대로 안 이뤄져…총 6720만원 '철퇴'
'계정 무단 공유' 등 30개 지자체에도 시정조치 권고 처분

< 위반 사업자 행정처분 내용 >

(단위 : 만원)

사업자명	위 반 내 용	시정명령	과징금	과태료
㈜네이처리퍼블릭	• 개인정보 보호조치 위반(§28③)	○	2,120	1,000
㈜에스디생명공학	• 개인정보 유출통지 위반(§27조의3①)	○	-	500
	• 개인정보 보호조치 위반(§28③)		850	800
테슬라코리아㈜	• 개인정보 유출통지 위반(§27조의3①)	○	-	500
㈜씨트립코리아	• 개인정보 유출신고 위반(§27조의3①)	○	-	500



암호화 등 보호조치 제대로 안 이뤄져…총 6720만원 '철퇴'

개인정보위는 27일 제2회 전체회의를 열고 네이처리퍼블릭, 에스디생명공학, 테슬라코리아, 씨트립코리아 등 4개 사업자에 대한 시정명령을 내리고 2970만원의 과징금과 3300만원의 과태료 부과를 의결했다. 이번 조사는 개인정보 유출 신고가 접수된 3개 사업자와 국민신문고로 민원이 제기된 1개 사업자에 대해 이뤄졌다.

네이처리퍼블릭은 미상의 해커가 'SQL 삽입(악성코드에 감염시킨 후 관리자 권한을 획득하는 방식) 공격' 으로 관리자 페이지에 로그인해 14건의 고객 정보가 유출된 것으로, 과징금 2120만원과 과태료 1000만원 처분을 받았다. 에스디생명공학도 해커로부터 쇼핑몰서비스 웹셸(web shell) 공격을 당해 쇼핑몰 회원정보가 유출돼 850만원의 과징금과 1300만원의 과태료를 부고받았다.

김진해 개인정보위 대변인은 "네이처리퍼블릭은 정보통신망법 제28조 제1항에 따른 개인정보 처리시스템에 대한 접근통제, 암호화 등 개인정보 보호조치를 하지 않아 과징금과 과태료가 부과됐다"고 설명했다.

- 상용화를 위해서는 **개인정보 암호화와 시큐어 코딩 인가**를 받아야 하기 때문에 시큐어 코딩은 중요하다.

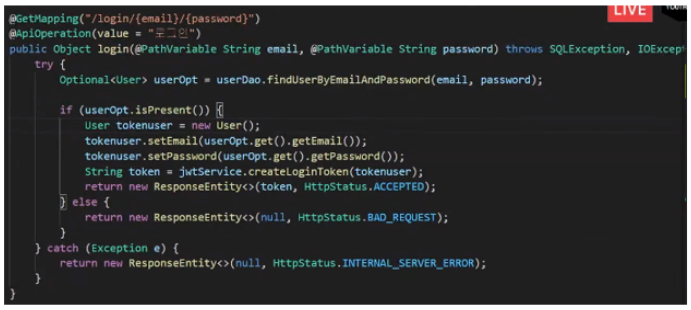
- 우리나라에서는 KISA(한국인터넷진흥원)에서 인가를 받아야한다.
- 외국에는 **OWASP(Open Web Application Security Project)**라는 비영리 단체가 있는데, 여기서 가장 빈번하고 크리티컬한 취약점 유형 TOP 10을 정하고 있다.

The Top 10 OWASP vulnerabilities in 2020 are:

- Injection
- Broken Authentication
- Sensitive Data Exposure
- XML External Entities (XXE)
- Broken Access control
- Security misconfigurations
- Cross Site Scripting (XSS)
- Insecure Deserialization
- Using Components with known vulnerabilities
- Insufficient logging and monitoring

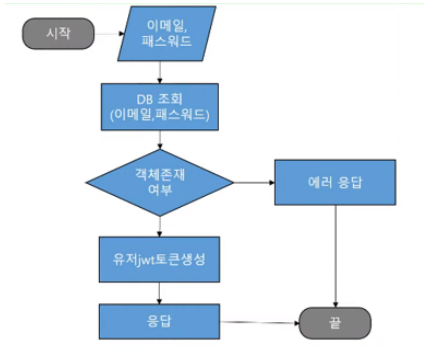
- 몇 년째 부동의 1위를 지키고 있는 유형은 **SQL Injection**이다.

2-1. 예시 코드



```
@GetMapping("/login/{email}/{password}")
@ApiOperation(value = "로그인")
public Object login(@PathVariable String email, @PathVariable String password) throws SQLException, IOExcep
    try {
        Optional<User> userOpt = userDao.findUserByEmailAndPassword(email, password);

        if (userOpt.isPresent()) {
            User tokenuser = new User();
            tokenuser.setEmail(userOpt.get().getEmail());
            tokenuser.setPassword(userOpt.get().getPassword());
            String token = jwtService.createLoginToken(tokenuser);
            return new ResponseEntity<>(token, HttpStatus.ACCEPTED);
        } else {
            return new ResponseEntity<>(null, HttpStatus.BAD_REQUEST);
        }
    } catch (Exception e) {
        return new ResponseEntity<>(null, HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```



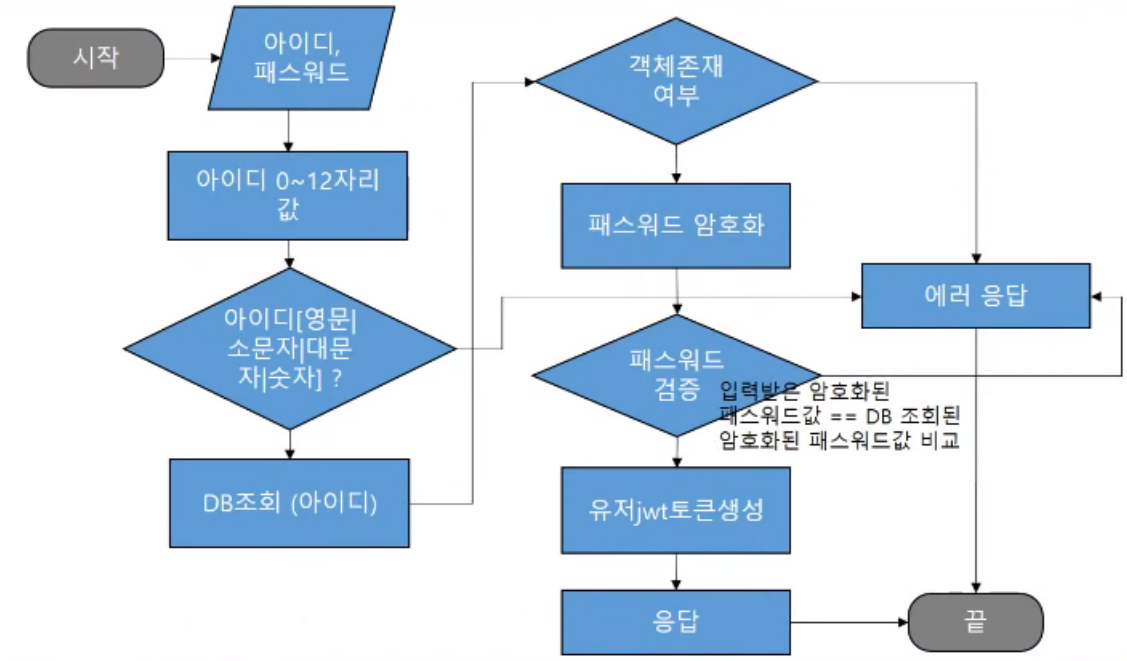
- 로그인 로직을 구현하는 코드의 예시다. 오른쪽 flowchart와 같은 로직 흐름을 보이고 있다.

```
select
    user0_.uid as uid1_11_,
    user0_.check_type as check_ty2_11_,
    user0_.clocation as clocatio3_11_,
    user0_.cphone as cphone4_11_,
    user0_.create_date as create_d5_11_,
    user0_.email as email6_11_,
    user0_.imgurl as imgurl7_11_,
    user0_.name as name8_11_,
    user0_.nickname as nickname9_11_,
    user0_.password as passwor10_11_
from
    user user0
where
    user0_.email=''OR 1=2 -- '
    and user0_.password='' UNION SELECT * FROM user ORDER BY 1 LIMIT 1 -- '
```

uid1_11_	check_ty2_11_	clocatio3_11_	cphone4_11_	create_d5_11_	email6_11_	imgurl7_11_	name8_11_	nickname9_11_	passwor10_11_
1	(NULL)	(NULL)	(NULL)	(NULL)	fedora3	(NULL)	관리자	관리자	passwd

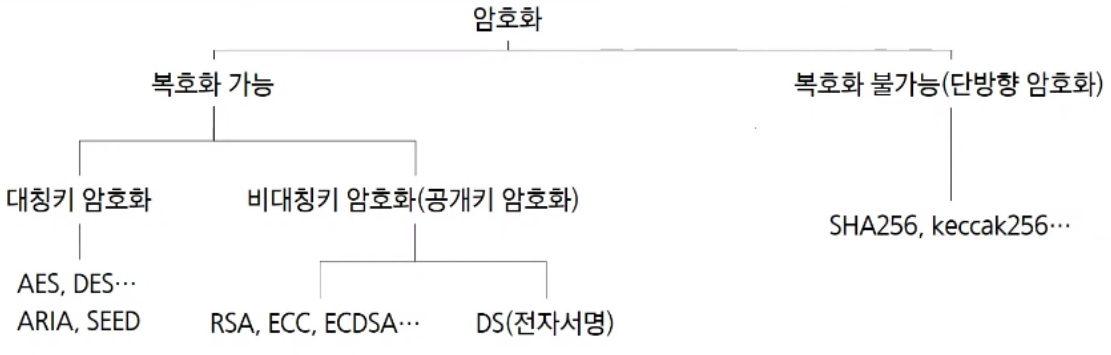
- 하지만 위와 같이 코드를 짜면 외부에서 악의적으로 쿼리문을 보내서 유출되어서는 안되는 **데이터를 탈취할 수도 있다.**
- 이를 막기 위해서는 **외부에서 들어오는 데이터 값들을 철저하게 검증**해야한다.

2-2. SQL Injection을 막기 위한 로직 수정



- **아이디의 길이 수를 제한**하는 것은 외부 공격을 막는 방법 중 하나인데, 그 이유는 보통 조작을 위해서 입력된 문자열의 길이는 길기 때문이다. 이렇게 길이 수를 제한하는 경우에는 프론트 뿐만 아니라 백에서도 검증이 이루어져야한다.
- 선택된 문자만 입력할 수 있게 해주는 **화이트리스트**를 활용하는 것도 보안을 높이는 방법이다.
- 입력 받은 패스워드는 **해시함수에 의해 암호화**가 되어 DB에서 조회된 암호화 된 패스워드 값과 비교를 하는데, 같으면 로그인 성공이고 아니면 에러를 리턴한다.
- **개인정보 암호화**는 정부 행정 지침이기 때문에 꼭 이루어져야하며, OWASP TOP 10에서 3위에 해당하는 Sensitive Data Exposure와도 관련 있는 내용이다.

3. 암호화



- 암호화 방식에는 크게 **복호화 가능 암호화**와 **복호화 불가능한 암호화(단방향 암호화)**가 있다.
- 복호화 불가능한 암호화에는 가장 대표적으로 **해시 함수**가 있으며, 실무에서 많이 사용되는 종류로는 **SHA256, keccak256** 등이 있다.

3-1. SHA256 암호화 예시

```
Hello SSafy!
-> BF21D9C9B5EE25E28C7EADA0F9C709FEC09016A8E0AC3F695530C1B1E205E4C0

Hello SSafy!
-> BF21D9C9B5EE25E28C7EADA0F9C709FEC09016A8E0AC3F695530C1B1E205E4C0

Hel1o SSafy!
-> 67FD612D914172387FCCEA7B87013779A732D6ECA794299596E7691C588D4E5F
```

- `Hello SSafy!`와 `Hel1o SSafy!`를 SHA256으로 해시 암호화한 모습은 굉장히 다르다.
- 복호화가 불가능하기 때문에 **검증을 할 때 사용**한다. 우리가 비밀번호를 잊어버려서 비밀번호 찾기를 할 때 기존의 비밀번호를 알려주는 것이 아닌 새로운 비밀번호를 설정하게 하는 것도 암호화된 비밀번호의 복호화가 불가능하기 때문이다.

3-2. 보안을 높이기 위해 바뀐 예시 코드의 모습

```
/***************/
@GetMapping("/newlogin/{userid}/{password}")
@ApiOperation(value = "로그인")
public Object newLogin(@PathVariable String userId, @PathVariable String password) throws SQLException, IOException {

    String newUserId = userId.substring(0,12);
    if (!java.util.regex.Pattern.compile("[a-zA-Z0-9]+").matcher(newUserId).matches()) return null;
    Optional<User> userOpt = userDao.findByName(newUserId);

    if (userOpt.isPresent()) {
        if (!userOpt.get().getPassword().equals(SHA256(password))) return null;
        User tokenuser = new User();
        tokenuser.setEmail(userOpt.get().getEmail());
        tokenuser.setPassword(userOpt.get().getPassword());
        String token = jwtService.createLoginToken(tokenuser);
        return new ResponseEntity<>(token, HttpStatus.ACCEPTED);
    } else {
        return new ResponseEntity<>(null, HttpStatus.BAD_REQUEST);
    }

}

private String SHA256(String password) {
    // encrypt
    return "";
}
/*********************/
```

4. 좋은 코딩 습관

1. 스트링 비교 시 `string.equals("")` 대신에 `"".equals(string)`을 사용한다. 아래의 예시에서 윗줄보다는 아랫줄과 같이 쓰는 것이 Null exception 에러를 방지할 수 있다.

```
request.getCheckType().equals("business"))
"business".equals( request.getCheckType() );
```

2. `@RequestBody` 바인딩 되는 부분을 JPA Entity 객체로 받으면 안된다. 대신에 Map을 사용해서 아래와 같이 받는다.

```
@PostMapping("/kakaologin")
@ApiOperation(value = "카카오 로그인")
public Object viewInfo(@RequestBody Map<String, Object> request) throws SQLException, IOException {
    String email = request.get("email").toString();
```

5. 요약

1. OWASP top 10의 취약점 1위로서 시스템 DB에 쿼리문을 주입하는 방법으로 **SQL Injection**이라고 한다.
2. 복호화 불가능 (단방향) 암호화에 쓰이는 SHA256이나 keccak256을 **해시 함수**라고 한다.
3. SSAFY에서 사용하는 정적 분석 툴로서 보안 취약점이나 버그 등을 리포트해주는 툴을 **Sonarqube**라고 한다.
4. 변수 String 타입의 id값이 null이어도 에러가 나지 않는 구문은 id.equals('TESTID')가 아닌 **'TESTID'.equals(id)**이다.

6. Q&A

1. SHA256으로 암호화했을 때 다이제스트를 역으로 추적할 수 있는데 추가적으로 난수형 salt를 생성해줘야 하나요?
    - SHA256으로 했을때는 여태 알려진 바로는 역으로 추척할 수가 없지만, 그 이외의 약한 해시함수들 MD5 나 SHA-0 SHA-1등을 생성 시 salt를 넣어주면 더 강한 암호화가 됩니다.

2. 해시함수는 공백 한 칸과 두 칸이 다르게 결과값이 나오나요?
    - 네 다르게 나옵니다. 그래서 원본 값 등을 검사하는 무결성등을 검사할 때 많이 쓰입니다.

3. 일반적으로 해시 처리를 할 때 해시충돌 가능성을 염두해두고 처리를 하나요?
    - 보통 SHA256, (SHA3계열)keccak256 등은 알려진 바로 해시충돌은 없습니다. 하지만 MD5나 SHA-0 SHA-1 등은 해시충돌이 밝혀졌기 때문에 실무에서 사용하지 않습니다. 더 염려가 되면 시간과 비용이 조금 더 드는 256비트 이상인 512정도로 하시면 됩니다.

4. Java에서 null 방지 꿀팁이 또 있을까요?
    - 자바8 이상부터는 Optional이라는 것이 생겼습니다. 그것을 적극 활용하시면 null Exception을 회피 하실수 있으실 것입니다.

5. SQL 인젝션은 매년 OWASP 보안취약점 1위로 선정되던데 매직쿼터를 막아주면 어떤 방법으로 공격하나요?
    - 싱글쿼터를 말씀하시는거 같은데, 그 이외에도 주석 및 논리부정 등으로 공격할 수 있습니다.