# 암호화 및 CSRF

WHATEVER YOU WANT, MAKE IT REAL.

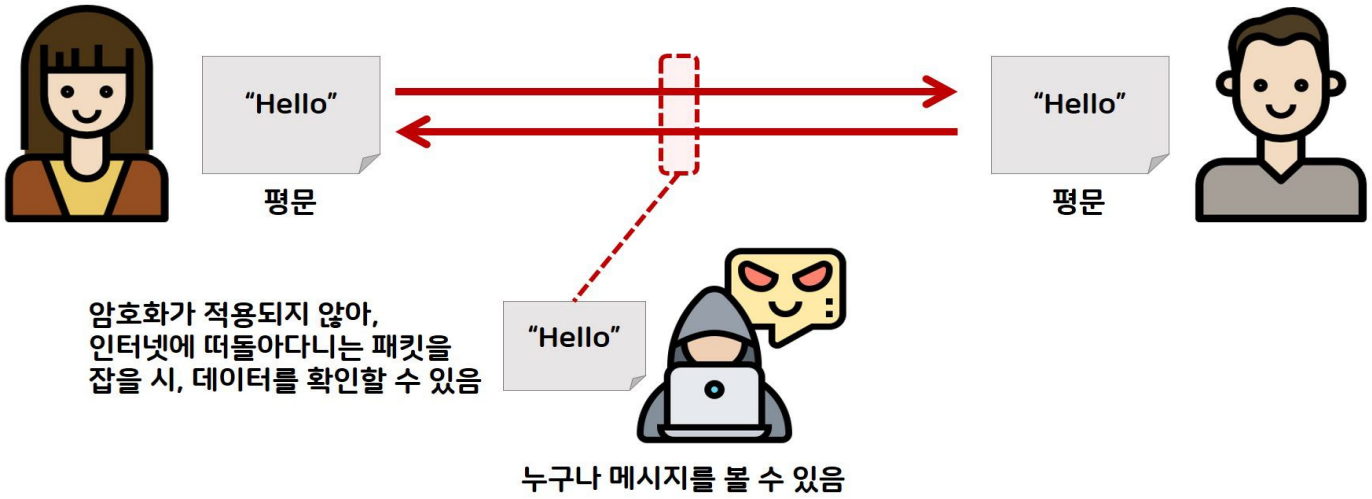
강사 허정식

목표 !암호화 및 CSRF

{ 암호화 모듈 및 CSRF }

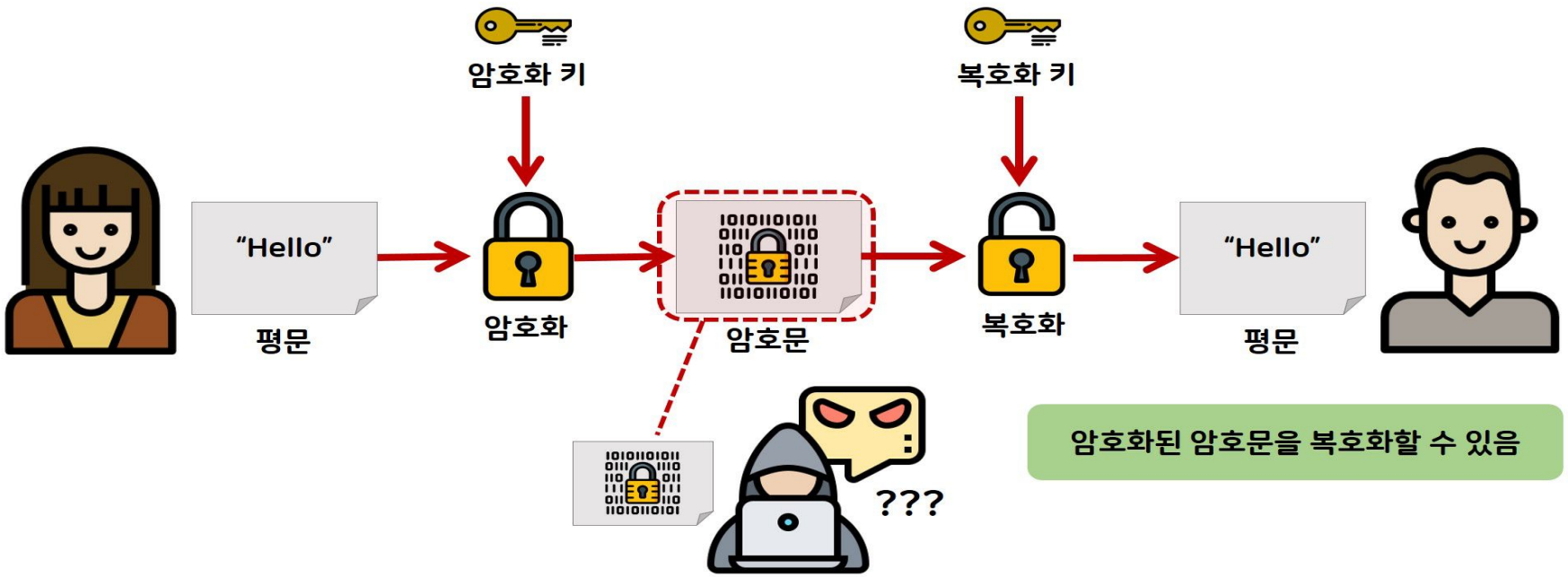
# 암호화

암호화 란? 사용자가 입력한 데이터를 알아볼 수 없는 데이터로 변경하는 과정

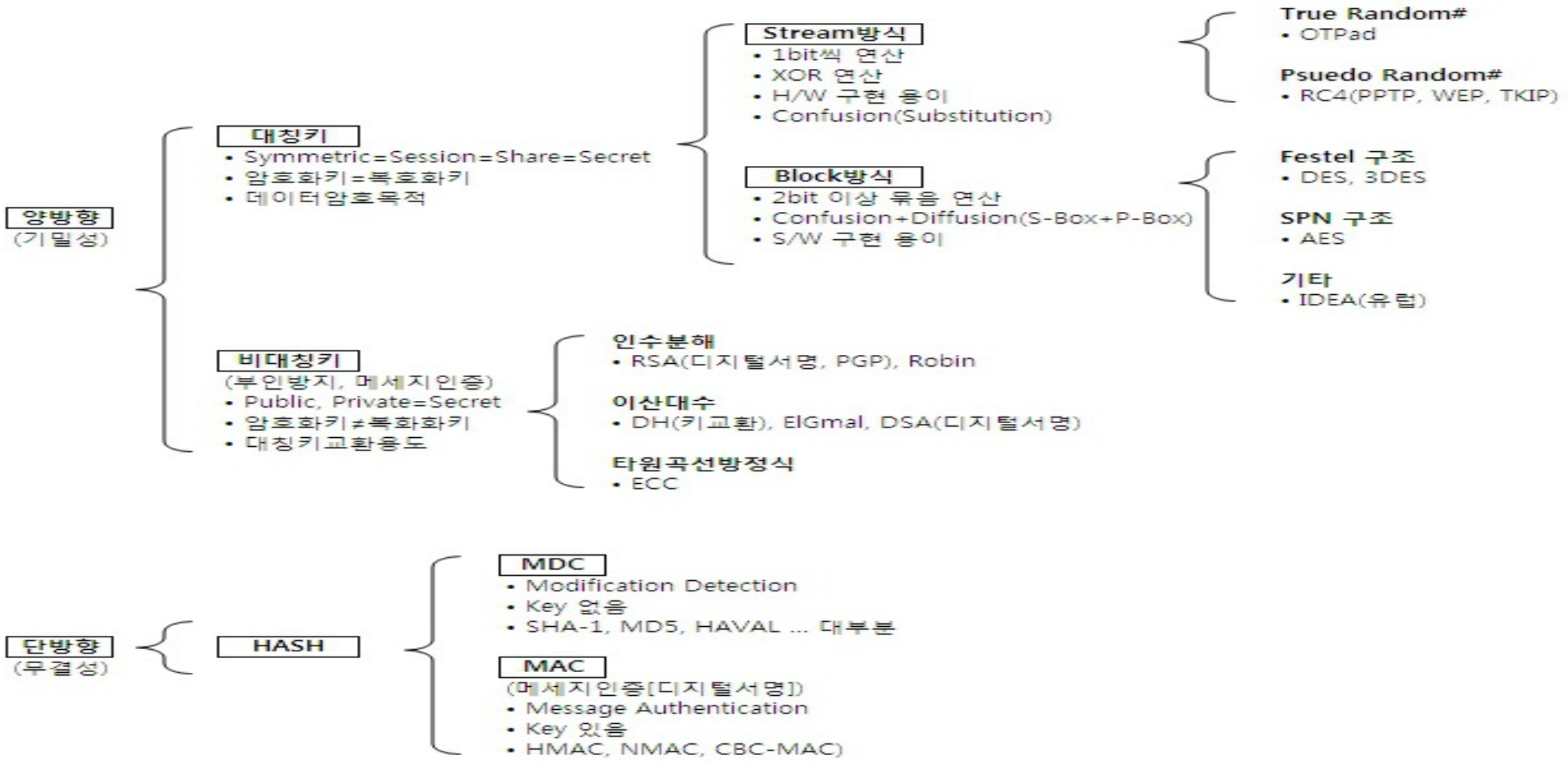


# 복호화

복호화 란?　　복호화(Decrypt)는 암호화된 데이터를 정상적인 데이터로 변경하는 과정



# 암호화 알고리즘 및 종류


양방향
(기밀성)
대칭키
• Symmetric=Session=Share=Secret
• 암호화키=복호화키
• 데이터암호목적
Stream방식
• 1bit씩 연산
• XOR 연산
• H/W 구현 용이
• Confusion(Substitution)
True Random#
• OTPad
Psuedo Random#
• RC4(PPTP, WEP, TKIP)
Block방식
• 2bit 이상 묶음 연산
• Confusion+Diffusion(S-Box+P-Box)
• S/W 구현 용이
Festel 구조
• DES, 3DES
SPN 구조
• AES
기타
• IDEA(유럽)
비대칭키
(부인방지, 메세지인증)
• Public, Private=Secret
• 암호화키≠복화화키
• 대칭키교환용도
인수분해
• RSA(디지털서명, PGP), Robin
이산대수
• DH(키교환), ElGmal, DSA(디지털서명)
타원곡선방정식
• ECC
단방향
(무결성)
HASH
MDC
• Modification Detection
• Key 없음
• SHA-1, MD5, HAVAL ... 대부분
MAC
(메세지인증[디지털서명])
• Message Authentication
• Key 있음
• HMAC, NMAC, CBC-MAC)

# 스프링 시큐리티 제공 - 암호화 객체들

PasswordEncoder passwordEncoder;

| Class Summary | |
|---|---|
| BaseDigestPasswordEncoder | Convenience base for digest password encoders. |
| BasePasswordEncoder | Convenience base for all password encoders. |
| LdapShaPasswordEncoder | A version of ShaPasswordEncoder which supports Ldap SHA and SSHA (salted-SHA) encodings. |
| Md4PasswordEncoder | MD4 implementation of PasswordEncoder. |
| Md5PasswordEncoder | MD5 implementation of PasswordEncoder. |
| MessageDigestPasswordEncoder | Base for digest password encoders. |
| PlaintextPasswordEncoder | Plaintext implementation of PasswordEncoder. |
| ShaPasswordEncoder | SHA implementation of PasswordEncoder. |

https://docs.spring.io/spring-security/site/docs/3.1.6.RELEASE/apidocs/org/springframework/security/authentication/encoding/package-summary.html

# 스프링 시큐리티 제공 - 암호화 객체

현업에서 가장 많이 쓰는 객체

**BCryptPasswordEncoder 란?**

BCryptPasswordEncoder는 BCrypt 해싱 함수(BCrypt hashing function)를 사용해서 비밀번호를 인코딩

encode() - Stirng 형태로 넘어온 매개변수를 Bcrypt 방식으로 인코딩 한다.

matches() - 인코딩 전 패스워드와 인코딩 후 패스워드를 비교하여 boolean 형태로 값을 리턴해준다.

# 스프링 시큐리티 제공 - 암호화 객체 예제

**BCryptPasswordEncoder** passwordEncoder;

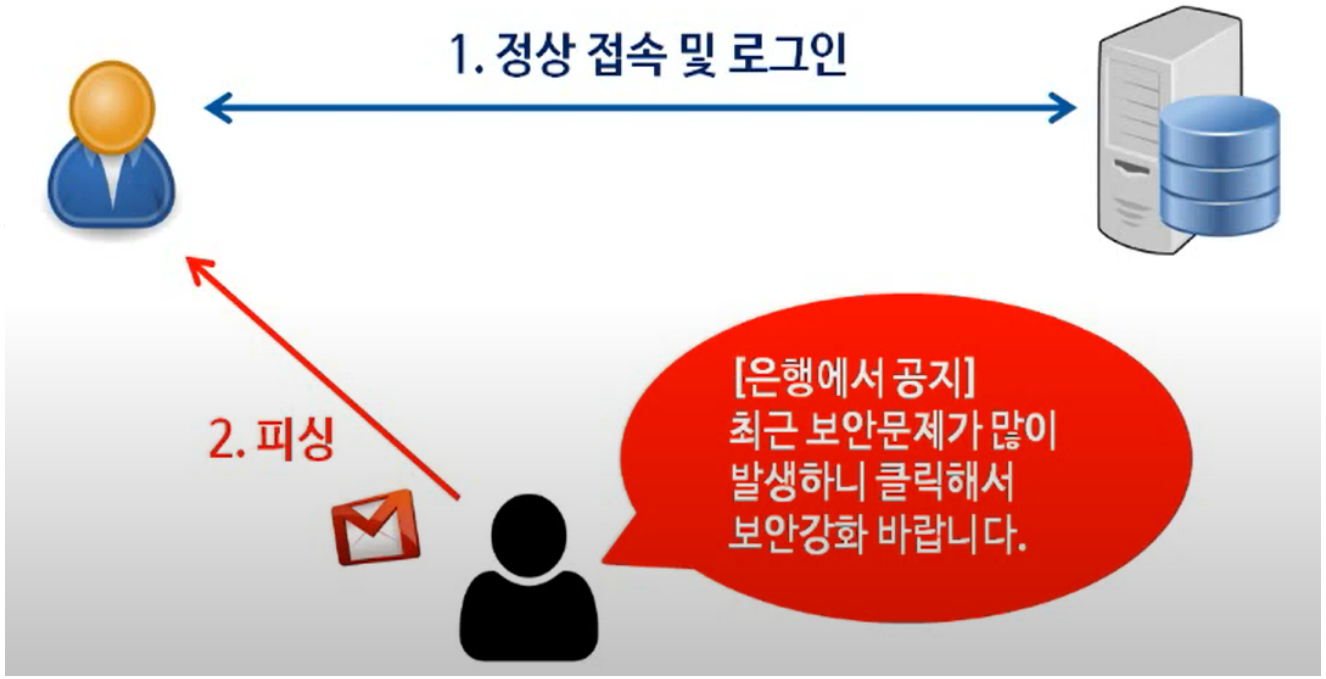
```
@SpringBootTest
public class EncoderTest{

        @Autowired
        private BCryptPasswordEncoder passwordEncoder;

        @Test
        void encryptTest(){
                String plainPassword = "1231231231231";
                String encodedPassword = passwordEncoder.encode(plainPassword);

                System.out.println("==================== EncryptPassword ==============
======");
    System.out.println(encodePassword);
    System.out.println("==================== EncryptPassword ====================");

                assertAll(
            () -> assertNotEquals(plainPassword, encodePassword),
            () -> assertTrue(passwordEncoder.matches(plainPassword, encodePassword))
   );
        }
}
```

# CSRF란?

Cross Site Request Forgery

사이트간 요청 위조

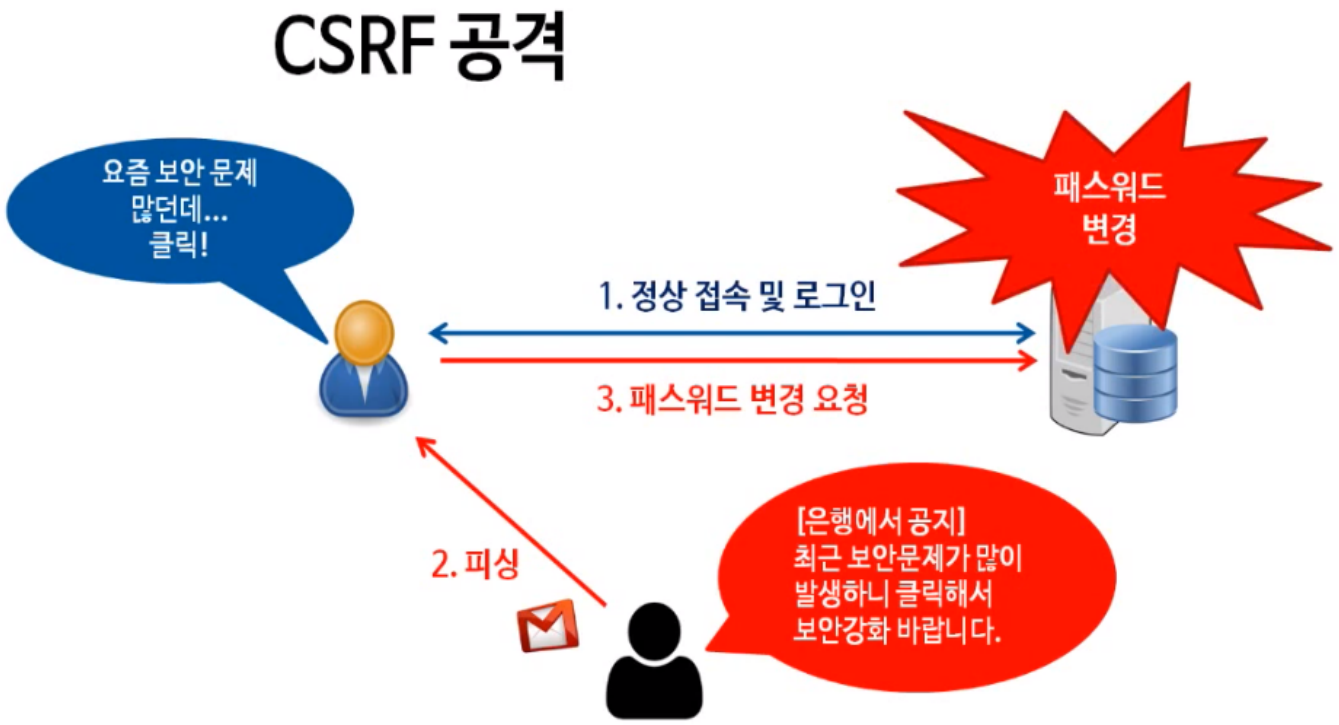
피싱을 활용해 사용자 모르게 해스워드 변경
옥션 해킹 사건에서 사용된 공격 기법




개념도 중요하지만 더 중요한 건 **사용자가 로그인되어있는 상태여야한다**라는 조건

# CSRF 의 이해

그 유명한 옥션 탈취 사건




개념도 중요하지만 더 중요한 건 **사용자가 로그인되어있는 상태여야한다**라는 조건

# CSRF 의 이해

<img src="http://auction.com/changeUserAcoount?id=admin&password=admin" width="0" height="0">

1.옥션 관리자 중 한명이 관리 권한을 가지고 회사내에서 작업을 하던 중 메일을 조회한다. (로그인이 이미 되어있다고 가정하면 관리자로서의 유효한 쿠키를 갖고있음)

2.해커는 위와 같이 태그가 들어간 코드가 담긴 이메일을 보낸다. 관리자는 이미지 크기가 0이므로 전혀 알지 못한다.

3.피해자가 이메일을 열어볼 때, 이미지 파일을 받아오기 위해 URL이 열린다.

4.해커가 원하는 대로 관리자의 계정이 id와 pw 모두 admin인 계정으로 변경된다.

위와같은 방법을 실행하려면 두가지 조건이 선행되어야한다.

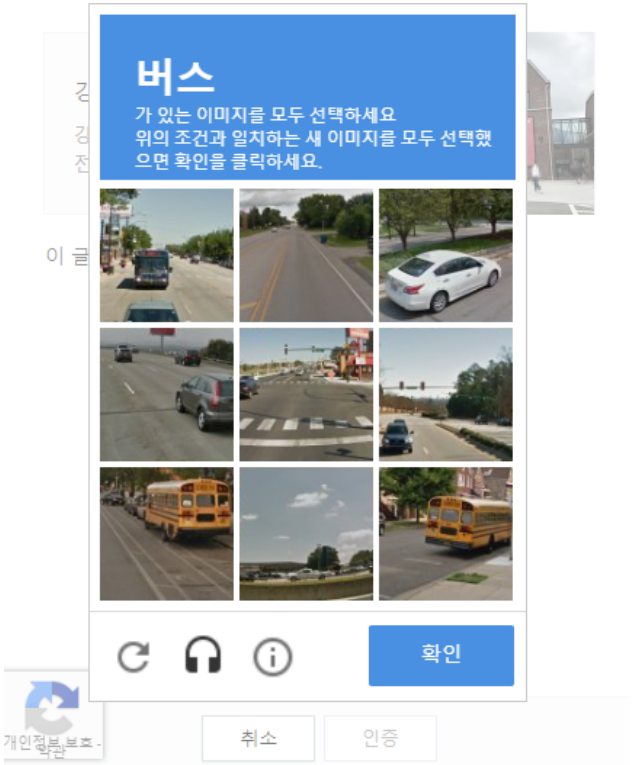
위조 요청을 전송하는 서비스에 희생자가 로그인 상태
희생자가 해커가 만든 피싱 사이트에 접속

# CSRF 방어방법

## 1. CAPTCHA사용

1.회원정보 변경, 게시글 작성 등 CSRF의 공격을 막아야 하는 페이지에서 캡챠를 사용해 인증이 가능한 것만 요청을 처리해주는 방법이 있다.

2.캡챠는 랜덤 이미지를 통해 인증이 되므로 사용자 몰래 요청하는것이 불가능 하다는 장점이 있다.




## 2. CSRF 토큰 사용 (Spring Security에서 지원)

토큰을 생성해 그 값을 비교하여 토큰이 존재하지 않거나 다른 경우 동작을 허용하지 않는 방법

# 스프링 시큐리티 CSRF 토큰 사용 방법

1. GET 방식 이외에 PUT,POST ,UPDATE 에서 적용됨

```
<form action="/login" method='post'>
    아이디 : <input type="text" name="username" value="admin"><br/>
    비밀번호 : <input type="text" name="password" value="admin"><br/>
    <input type="submit" value="로그인하기">
    <input type="hidden" name="${_csrf.parameterName }" value="${_csrf.token }"/>
    <!-- ㄴ> 히든을 빼면 로그인이 안됨 중요중요-->
</form>
```

2. CSRF 토큰 사용 (Spring Security에서 지원)

```
<form action="/login" method="post"> == $0
  " 아이디 : "
  <input type="text" name="username" value="admin">
  <br>
  " 비밀번호 : "
  <input type="text" name="password" value="admin">
  <br>
  <input type="submit" value="로그인하기">
  <input type="hidden" name="_csrf" value="b64c20c1-2e99-4bcd-8a03-7e3702dc5571">
  <!-- ㄴ> 히든을 빼면 로그인이 안됨 중요중요-->
</form>
```

csrf 토큰은 매번 value값이 바뀌고, hidden으로 포함되어 들아가기 때문에 공격자 입장에서는 고정된 쿼리문만 전송해서는 더 이상 명령이 작동하지 않고 매번바뀌는 csrf토큰값을 그때그때 찍어서 맞춰야함 사실상 완전방어