



# 주니어 개발자를 위한, 안전한 소프트웨어 만들기

문성훈

2025. 05. 17.

# 발표자 소개

문성훈

가비아

클라우드 개발팀 / 클라우드 서비스 API 개발



# 목차

- 개발자가 보안을 공부해야 하는 이유
- 안전한 소프트웨어를 만드는 방법
- 보안적 사고방식 기르기 🍀

개발자가 보안을 공부해야 하는 이유



# 개발자가 보안을 공부해야 하는 이유


## 개발자의 직업 윤리

SBS NEWS

PICK ⓘ

### URL에 접수번호 바꾸자...13년치 대학 지원자 정보 술술

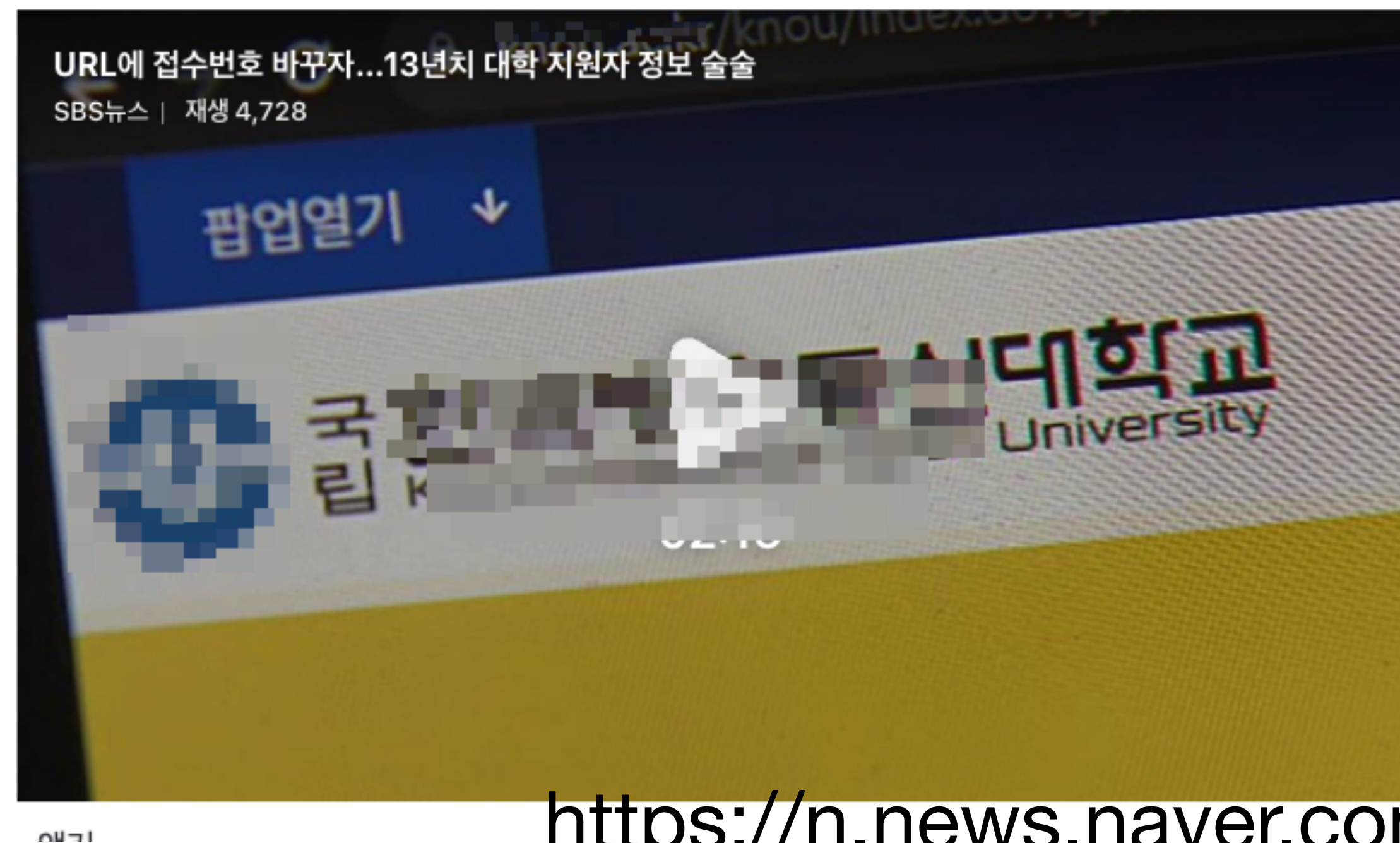
입력 2023.12.05. 오후 9:04 · 수정 2023.12.05. 오후 11:46 기사원문

 최승훈 기자 TALK

 21

 12



<https://n.news.naver.com/article/055/0001111825>

# 개발자가 보안을 공부해야 하는 이유

## SW 개발 단계별 결함 수정비용 분석

구분	설계단계	코딩단계	통합단계	베타제품	제품출시
설계과정 결함	1배	5배	10배	15배	30배
코딩과정 결함		1배	10배	20배	30배
통합과정 결함			1배	10배	20배

The Economic Impacts of Inadequate Infrastructure for SW Testing(2002.5, NIST)

# 안전한 소프트웨어를 만드는 방법

# 안전한 소프트웨어를 만드는 방법

## OWASP Top 10

OWASP Top 10 – 2013
A1 – Injection
A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References
A5 – Security Misconfiguration
A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control
A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards

OWASP Top 10 – 2017
A1 – Injection
A2 – Broken Authentication
A3 – Sensitive Data Exposure
A4 – XML External Entities (XXE)
A5 – Broken Access Control
A6 – Security Misconfiguration
A7 – Cross-Site Scripting (XSS)
A8 – Insecure Deserialization
A9 – Using Components with Known Vulnerabilities
A10 – Insufficient Logging & Monitoring

OWASP Top 10 – 2021
A1 – Broken Access Control
A2 – Cryptographic Failures
A3 – Injection
A4 – Insecure Design
A5 – Security Misconfiguration
A6 – Vulnerable and Outdated Components
A7 – Identification and Authentication Failures
A8 – Software and Data Integrity Failures
A9 – Security Logging and Monitoring Failures
A10 – Server-Side Request Forgery



# 안전한 소프트웨어를 만드는 방법

## OWASP Top 10

OWASP Top 10 - 2013	OWASP Top 10 - 2017	OWASP Top 10 - 2021
A1 - Injection	A1 - Injection	A1 - Broken Access Control
A2 - Broken Authentication and Session Management	A2 - Broken Authentication	A2 - Cryptographic Failures
A3 - Cross-Site Scripting (XSS)	A3 - Sensitive Data Exposure	A3 - Injection
A4 - Insecure Direct Object References	A4 - XML External Entities (XXE)	A4 - Insecure Design
A5 - Security Misconfiguration	A5 - Broken Access Control	A5 - Security Misconfiguration
A6 - Sensitive Data Exposure	A6 - Security Misconfiguration	A6 - Vulnerable and Outdated Components
A7 - Missing Function Level Access Control	A7 - Cross-Site Scripting (XSS)	A7 - Identification and Authentication Failures
A8 - Cross-Site Request Forgery (CSRF)	A8 - Insecure Deserialization	A8 - Software and Data Integrity Failures
A9 - Using Known Vulnerable Components	A9 - Using Components with Known Vulnerabilities	A9 - Security Logging and Monitoring Failures
A10 - Unvalidated Redirects and Forwards	A10 - Insufficient Logging & Monitoring	A10 - Server-Side Request Forgery

# 안전한 소프트웨어를 만드는 방법

## OWASP Top 10

OWASP Top 10 – 2013
A1 – Injection
A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References
A5 – Security Misconfiguration
A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control
A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards

OWASP Top 10 – 2017
A1 – Injection
A2 – Broken Authentication
A3 – Sensitive Data Exposure
A4 – XML External Entities (XXE)
A5 – Broken Access Control
A6 – Security Misconfiguration
A7 – Cross-Site Scripting (XSS)
A8 – Insecure Deserialization
A9 – Using Components with Known Vulnerabilities
A10 – Insufficient Logging & Monitoring

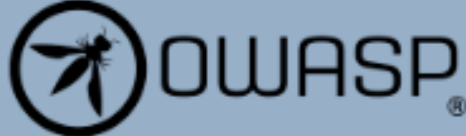
OWASP Top 10 – 2021
A1 – Broken Access Control
A2 – Cryptographic Failures
A3 – Injection
A4 – Insecure Design
A5 – Security Misconfiguration
A6 – Vulnerable and Outdated Components
A7 – Identification and Authentication Failures
A8 – Software and Data Integrity Failures
A9 – Security Logging and Monitoring Failures
A10 – Server-Side Request Forgery



OWASP Top Ten | OWASP Fou x +

← → ↺ owasp.org/www-project-top-ten/ 📄 ☆ 📷 🗑️ 📧 📧 📧 📧 📧 다시 실행하여 업데이트하세요. ⋮

Please support the OWASP mission to improve software security through open source initiatives and community education. [Donate Now!](#) ✕

 PROJECTS CHAPTERS EVENTS ABOUT 🔍 [Store](#) [Donate](#) [Join](#)

## OWASP Top Ten

[Main](#) Translation Efforts Sponsors Data 2025

### Important note:

#### OWASP Top Ten 2025

Current project status as of September 2024:

- We are planning to announce the release of the **OWASP Top 10:2025** in the first half of 2025.
- **Data Collection (Now - December 2024):** Please donate your application penetration testing statistics.

[Stay Tuned!](#)

---

The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications.

## Globally recognized by developers as the first step towards more secure coding.

Companies should adopt this document and start the process of ensuring that their web applications minimize these risks. Using the OWASP Top 10 is perhaps the most effective first step towards changing the software development culture within your organization into one that produces more secure code.

### Top 10 Web Application Security Risks


[Watch](#) 368 [Star](#) 1,203


**The OWASP® Foundation** works to improve the security of software through its community-led open source software projects, hundreds of chapters worldwide, tens of thousands of members, and by hosting local and global conferences.


---


#### Project Information

- [OWASP Top 10:2021](#)
- [Making of OWASP Top 10](#)
- [OWASP Top 10:2021 - 20th Anniversary Presentation \(PPTX\)](#)

 Flagship Project

 Documentation

 Builder

 Defender

- [Previous Version \(2017\)](#)

#### Downloads or Social Links

- [OWASP Top 10 2017](#)
- [Other languages](#) → tab 'Translation Efforts'

#### Social

[Twitter](#)

#### Code Repository



# 안전한 소프트웨어를 만드는 OWASP Top 10 - Broken Access Control

## Description

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits. Common access control vulnerabilities include:

- Violation of the principle of least privilege or deny by default, where access should only be granted for particular capabilities, roles, or users, but is available to anyone.
- Bypassing access control checks by modifying the URL (parameter tampering or force browsing), internal application state, or the HTML page, or by using an attack tool modifying API requests.
- Permitting viewing or editing someone else's account, by providing its unique identifier (insecure direct object references)
- Accessing API with missing access controls for POST, PUT and DELETE.
- Elevation of privilege. Acting as a user without being logged in or acting as an admin when logged in as a user.
- Metadata manipulation, such as replaying or tampering with a JSON Web Token (JWT) access control token, or a cookie or hidden field manipulated to elevate privileges or abusing JWT invalidation.
- CORS misconfiguration allows API access from unauthorized/untrusted origins.
- Force browsing to authenticated pages as an unauthenticated user or to privileged pages as a standard user.

## How to Prevent

Access control is only effective in trusted server-side code or server-less API, where the attacker cannot modify the access control check or metadata.

- Except for public resources, deny by default.
- Implement access control mechanisms once and re-use them throughout the application, including minimizing Cross-Origin Resource Sharing (CORS) usage.
- Model access controls should enforce record ownership rather than accepting that the user can create, read, update, or delete any record.
- Unique application business limit requirements should be enforced by domain models.
- Disable web server directory listing and ensure file metadata (e.g., .git) and backup files are not present within web roots.
- Log access control failures, alert admins when appropriate (e.g., repeated failures).
- Rate limit API and controller access to minimize the harm from automated attack tooling.
- Stateful session identifiers should be invalidated on the server after logout. Stateless JWT tokens should rather be short-lived so that the window of opportunity for an attacker is minimized. For longer lived JWTs it's highly recommended to follow the OAuth standards to revoke access.

Developers and QA staff should include functional access control unit and integration tests.

## Example Attack Scenarios

**Scenario #1:** The application uses unverified data in a SQL call that is accessing account information:

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

An attacker simply modifies the browser's 'acct' parameter to send whatever account number



# 안전한 소프트웨어를 만드는

## OWASP Top 10 - Broken Access Control

### Description

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits. Common access control vulnerabilities include:

- Violation of the principle of least privilege or deny by default, where access should only be granted for particular capabilities, roles, or users, but is available to anyone.
- Bypassing access control checks by modifying the URL (parameter tampering or force browsing), internal application state, or the HTML page, or by using an attack tool modifying API requests.
- Permitting viewing or editing someone else's account, by providing its unique identifier (insecure direct object references)
- Accessing API with missing access controls for POST, PUT and DELETE
- Elevation of privilege. Acting as a privileged user when only a standard user is logged in as a user.
- Metadata manipulation, such as replaying or tampering with a JSON Web Token (JWT) access control token, or a cookie or hidden field manipulated to elevate privileges or abusing JWT invalidation.
- CORS misconfiguration allows API access from unauthorized/untrusted origins.
- Force browsing to authenticated pages as an unauthenticated user or to privileged pages as a standard user.

### How to Prevent

Access control is only effective in trusted server-side code or server-less API, where the attacker cannot modify the access control check or metadata.

- Except for public resources, deny by default.
- Implement access control mechanisms once and re-use them throughout the application, including minimizing Cross-Origin Resource Sharing (CORS) usage.
- Model access controls should enforce record ownership rather than accepting that the user can create, read, update, or delete any record.
- Unique application business limit requirements should be enforced by domain models.
- Disable web server directory listing and ensure file metadata (e.g., .git) and backup files are not present within web roots.
- Log access control failures, alert admins when appropriate (e.g., repeated failures).
- Rate limit API and controller access to minimize the harm from automated attack tooling.
- Stateful session identifiers should be invalidated on the server after logout. Stateless JWT tokens should rather be short-lived so that the window of opportunity for an attacker is minimized. For longer lived JWTs it's highly recommended to follow the OAuth standards to revoke access.

Developers and QA staff should include functional access control unit and integration tests.

매개변수 변조 같이 URL을 수정하는 등, 접근 제어를 우회하는 행위.

### Example Attack Scenarios

**Scenario #1:** The application uses unverified data in a SQL call that is accessing account information:

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

An attacker simply modifies the browser's 'acct' parameter to send whatever account number

# 안전한 소프트웨어를 만드는 방법

## OWASP Top 10

### 소프트웨어 개발보안 가이드



# 안전한 소프트웨어를 만드는 방법

논리적 결함을 검증하자

# 안전한 소프트웨어를 만드는 방법

논리적 결함을 검증하자

**OpenSSL** Heartbleed 취약점  
Cryptography and SSL/TLS Toolkit



# 안전한 소프트웨어를 만드는 방법

논리적 결함을 검증하자

OpenSSL Heartbleed 취약점  
Cryptography and SSL/TLS Toolkit

```
1472 + if (1 + 2 + payload + 16 > s->s3->rrec.length)
1473 + return 0; /* silently discard per RFC 6520 sec. 4 */
```

# 안전한 소프트웨어를 만드는 방법

공급망 공격을 이해하자



어떤 직원이 만든 회사의 제품(소프트웨어)

해커

# 안전한 소프트웨어를 만드는 방법

공급망 공격을 이해하자



어떤 직원이 만든 회사의 제품(소프트웨어)



해커



# 안전한 소프트웨어를 만드는 방법

공급망 공격을 이해하자



우리가 사용하는 수많은 3rd-party....

```
"dependencies": {  
  "bcrypt": "^5.1.1",  
  "cls-rtracer": "^2.6.3",  
  "cross-env": "^7.0.3",  
  "dayjs": "^1.11.13",  
  "dotenv": "^16.4.7",  
  "express": "^4.21.1",  
  "express-async-errors": "^3.1.1",  
  "jsonwebtoken": "^9.0.2",  
  "knex": "^3.1.0",  
}
```



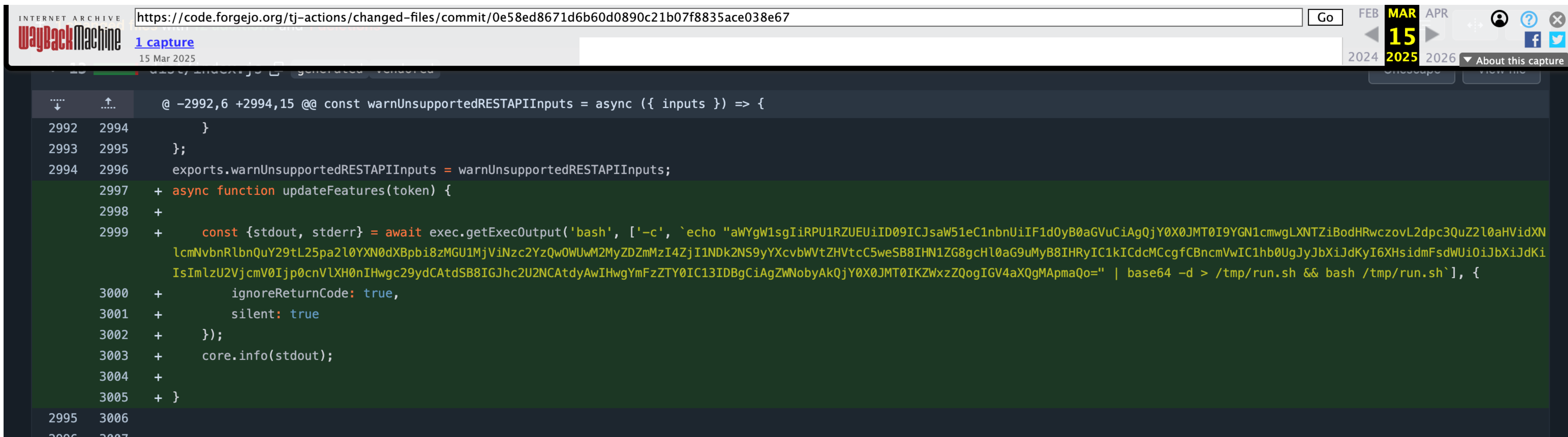
해커

어떤 직원이 만든 회사의 제품(소프트웨어)



# 공급망 공격을 이해하자

# Github Action인 changed-files 백도어 사건



# 안전한 소프트웨어를 만드는 방법

## 공급망 공격을 이해하자


버전 태그를 명시하면 안전한가?

```
with:
  python-version: 3.13

- name: Get Changed Files
  id: changed-files
  uses: tj-actions/changed-files@v45

# See:
```

Hacked 🥵



# 안전한 소프트웨어를 만드는 방법

## 공급망 공격을 이해하자

안전했던 케이스는 뭐지?

```
id: changed-files
```

```
uses: tj-actions/changed-files@aa08304bd477b800d468db44fe10f6c61f7f7b11 # v42.1.0
```

Secured 😊

# 보안적 사고방식 기르기

부제 : 창을 알아야 단단한 방패를 만들 수 있다.



# 보안적인 사고방식 기르기

취약점 != 눈에 보이는거

# 보안적인 사고방식 기르기

## 취약점 != 눈에 보이는거

### 비밀 홈페이지 🤫

관리자 이메일 주소

비밀번호

☐ 로그인 상태 유지

로그인

# 보안적인 사고방식 기르기

## 취약점 != 눈에 보이는거

### 비밀 홈페이지 🤫

관리자 이메일 주소

비밀번호

☐ 로그인 상태 유지

로그인

- SQL, Command 등의 인젝션 공격 안됨

# 보안적인 사고방식 기르기

## 취약점 != 눈에 보이는거

### 비밀 홈페이지 🤫

관리자 이메일 주소

비밀번호

☐ 로그인 상태 유지

로그인

- SQL, Command 등의 인젝션 공격 안됨
- URL Bruteforce도 안먹힘.



# 보안적인 사고방식 기르기

## 취약점 != 눈에 보이는거

### 비밀 홈페이지 🤫

관리자 이메일 주소

비밀번호

☐ 로그인 상태 유지

로그인

- SQL, Command 등의 인젝션 공격 안됨
- URL Bruteforce도 안먹힘.
- 로그인 계속 시도해도 막지 않네? -> **브루트포싱**

# 보안적인 사고방식 기르기

## 취약점 != 눈에 보이는거

### 비밀 홈페이지 🤫

관리자 이메일 주소

비밀번호

☐ 로그인 상태 유지

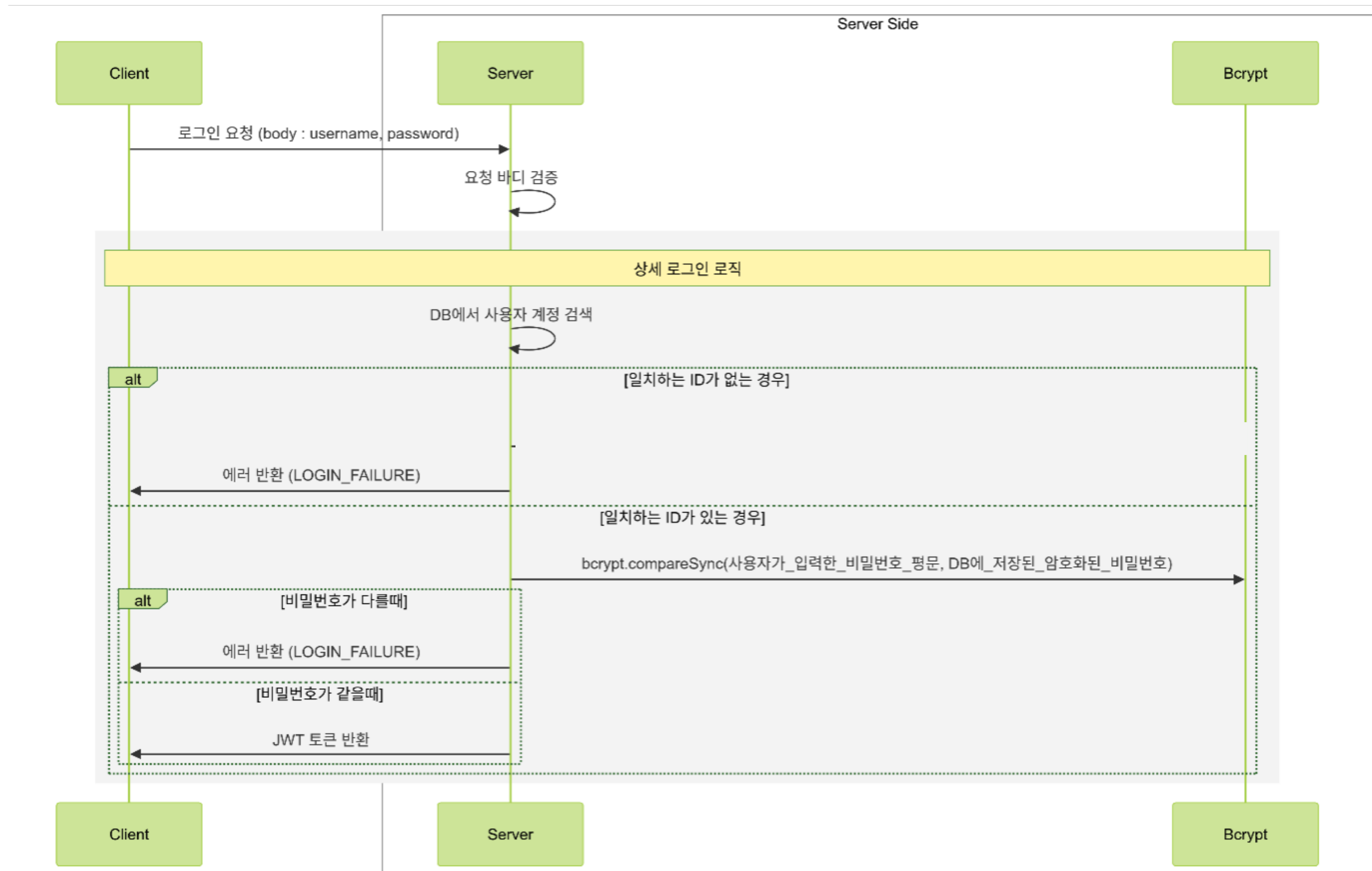
로그인

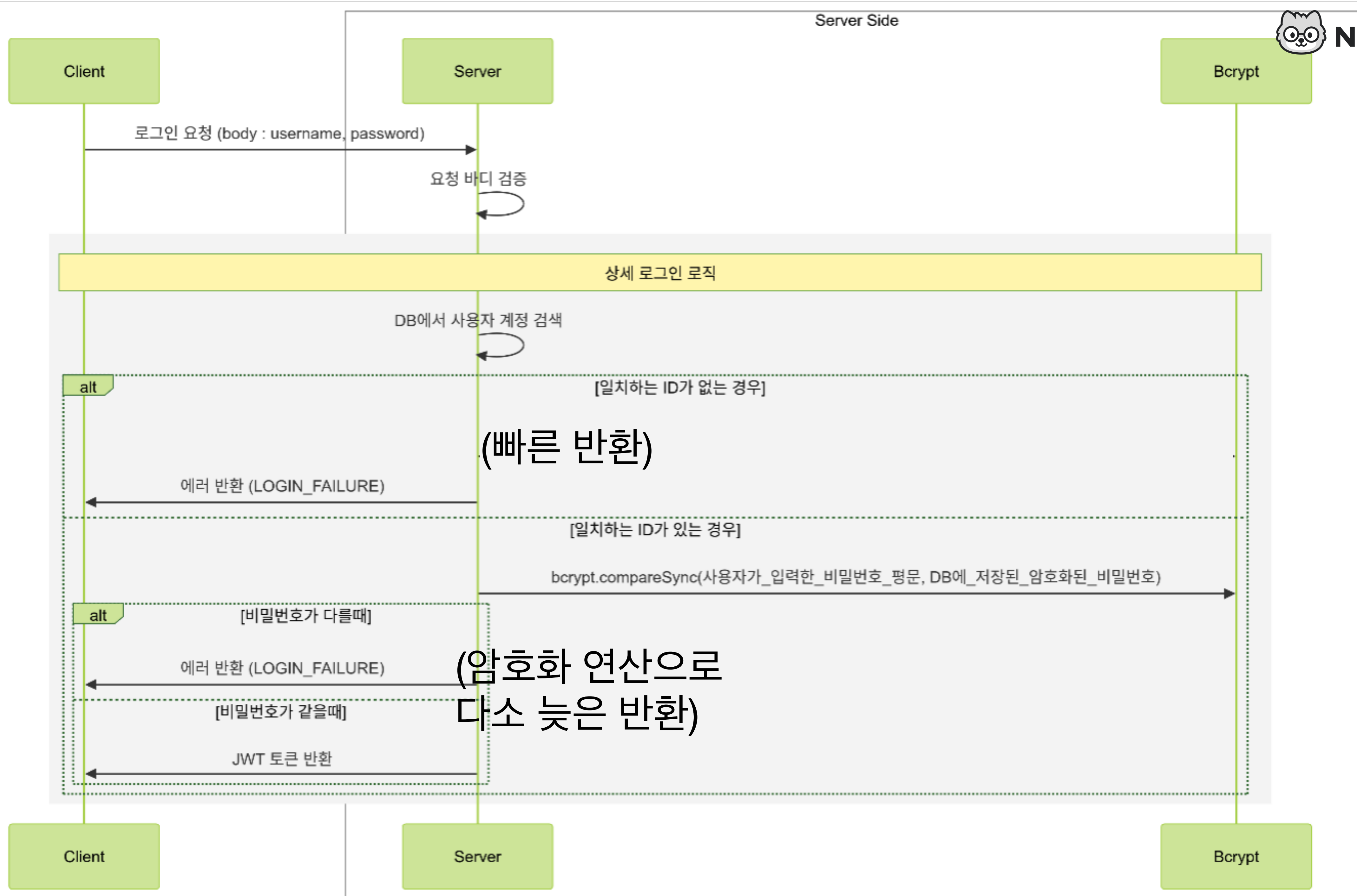
- SQL, Command 등의 인젝션 공격 안됨
- URL Bruteforce도 안먹힘.
- 로그인 계속 시도해도 막지 않네? -> 브루트포싱

**절망 포인트 :**  
근데... 브루트포싱은...  
가짓수가 너무많아....

# 보안적인 사고방식 기르기

취약점 != 눈에 보이는거







# 보안적인 사고방식 기르기

취약점 != 눈에 보이는거

== 존재하지 않는 아이디의 경우 ==

요청 1:	응답시간	0.0151초
요청 2:	응답시간	0.0147초
요청 3:	응답시간	0.0086초
요청 4:	응답시간	0.0095초
요청 5:	응답시간	0.0098초
요청 6:	응답시간	0.0052초
요청 7:	응답시간	0.0060초
요청 8:	응답시간	0.0088초
요청 9:	응답시간	0.0082초
요청 10:	응답시간	0.0081초

존재하지 않는 아이디의 경우 평균 응답시간: 0.0094초

== 존재하는 계정 (잘못된 비밀번호)의 경우 ==

요청 1:	응답시간	0.2205초
요청 2:	응답시간	0.2308초
요청 3:	응답시간	0.2291초
요청 4:	응답시간	0.2159초
요청 5:	응답시간	0.2156초
요청 6:	응답시간	0.2236초
요청 7:	응답시간	0.2225초
요청 8:	응답시간	0.2186초
요청 9:	응답시간	0.2132초
요청 10:	응답시간	0.2107초

존재하는 아이디(잘못된 비밀번호)의 경우 평균 응답시간: 0.2201초

# 보안적인 사고방식 기르기

취약점 != 눈에 보이는거

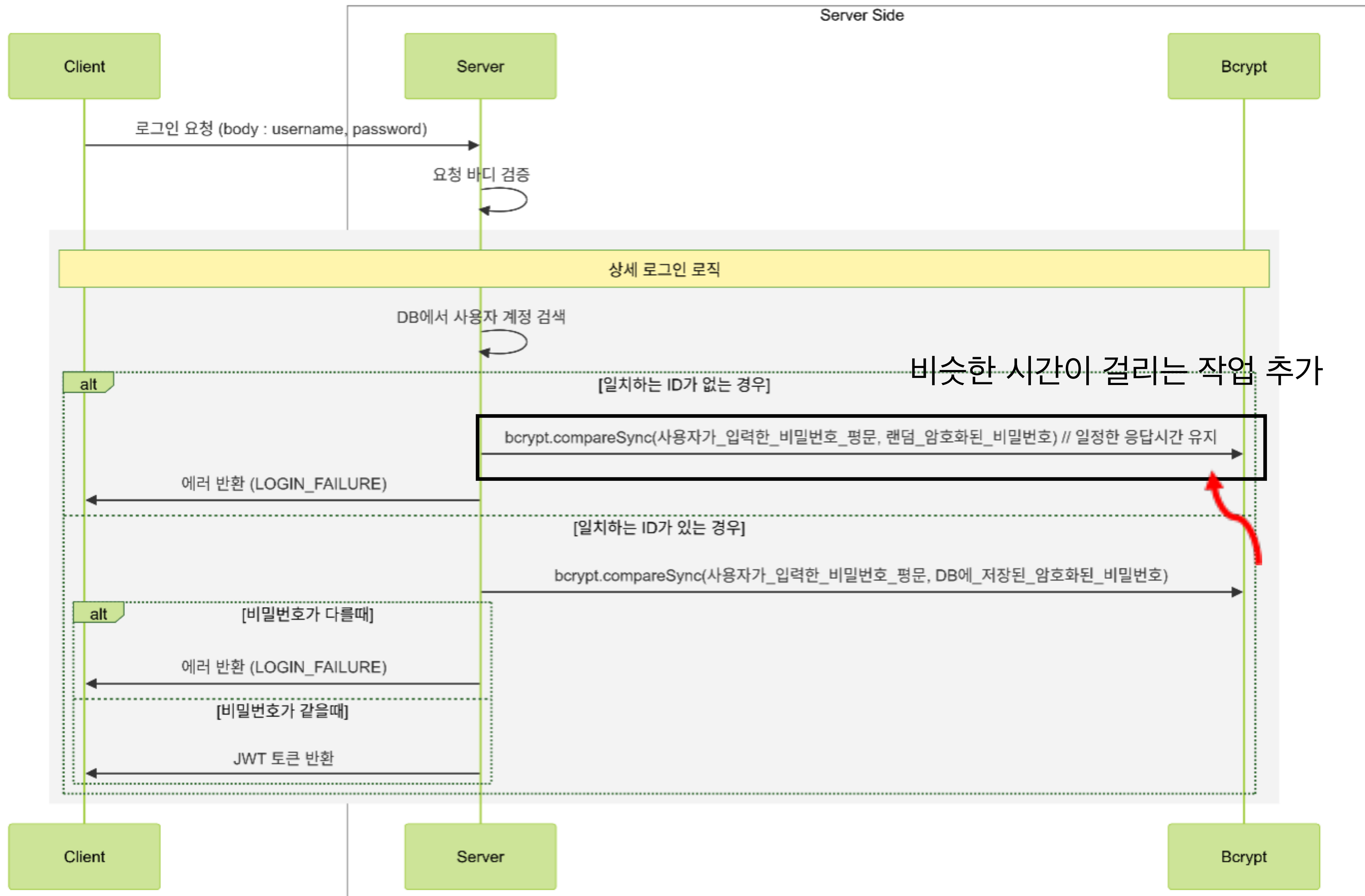
## 비밀 홈페이지 🤫

관리자 이메일 주소

비밀번호

☐ 로그인 상태 유지

로그인





# 보안적인 사고방식 기르기

## 취약점 != 눈에 보이는거

1

GitHub Advisory Database / GitHub Review / CVE-2012-5055

### Exposure of Sensitive Information to an Unauthorized Actor in Spring Security

Moderate severity GitHub Reviewed Published on May 17, 2022 to the GitHub Advisory Database • Updated on Jan 27, 2023

Vulnerability details Dependabot alerts 0

Package	Affected versions	Patched versions
org.springframework.security:spring-security-core (Maven)	< 2.0.8 >= 3.0.0, < 3.0.8 >= 3.1.0, < 3.1.3	2.0.8 3.0.8 3.1.3

**Description**

DaoAuthenticationProvider in VMware SpringSource Spring Security before 2.0.8, 3.0.x before 3.0.8, and 3.1.x before 3.1.3 does not check the password if the user is not found, which makes the response delay shorter and might allow remote attackers to enumerate valid usernames via a series of login requests.

**References**

- <https://nvd.nist.gov/vuln/detail/CVE-2012-5055>

**Severity**

Moderate

**EPSS score**

0.273% (68th percentile)

**Weaknesses**

► CWE-200

**CVE ID**

CVE-2012-5055

**GHSA ID**

GHSA-3533-rvpc-6x56

**Source code**

No known source code

This advisory has been edited. [See History.](#)

See something to contribute? [Suggest improvements for this vulnerability.](#)

Published to the GitHub Advisory Database on May 17, 2022

Reviewed on Jul 14, 2022

Last updated on Jan 27, 2023

사용자를 찾을 수 없는 경우 비밀번호를 확인하지 않으므로  
응답 지연이 더 짧아지고, 원격 공격자가 일련의 로그인 요청을 통해  
유효한 사용자 이름을 열거할 수 있습니다.

# 보안적인 사고방식 기르기

## 보안 솔루션은 다다익선?

Client-Side

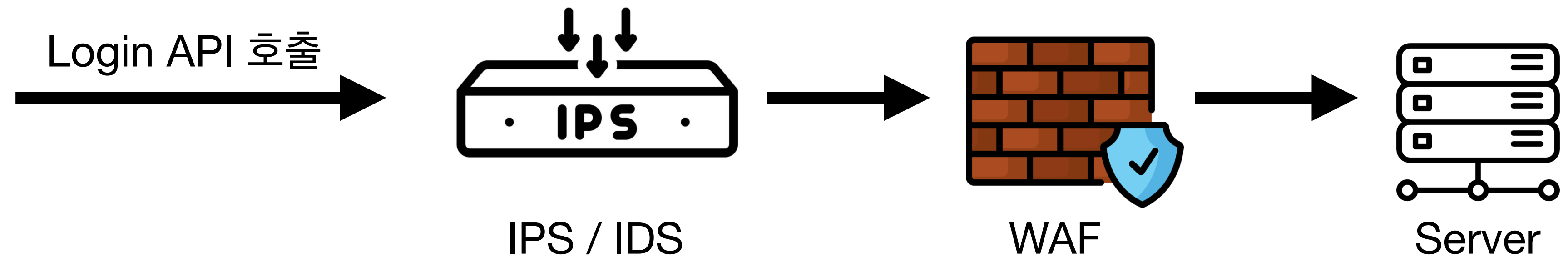
비밀 홈페이지 🤔

관리자 이메일 주소

비밀번호

☐ 로그인 상태 유지

로그인



# 보안적인 사고방식 기르기

## 해커들의 놀이터



dreamhack.io

<https://dreamhack.io>

### 해커들의 놀이터, Dreamhack

해커들의 놀이터, 드림핵 **Dreamhack**은 해커들의 놀이터입니다. ... 함께 공부하고 연습하며 지식을 나누고 실력을 향상할 수 있는 공간입니다.

#### 워게임

워게임 TOP 10 · 1 위. avatar. st4rlight. 소포모어. 29837 · 2 위 ...

#### 모든 CTF

모든 CTF · 전체 · 진행 중인 · 내가 참여한. Dreamhack CTF. 총 ...

#### Enterprise 소개

보안 분야를 잘 모르더라도 손쉽게 관리할 수 있어요 ...

#### 모든 로드맵

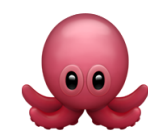
18354 명이 수강했어요. 난이도 쉬움. FREE. Sender. 난이도 쉬움 ...

#### CTF

1 위. avatar. keymoon. 프레시맨 · 2 위. G0RiyA. CTF 초보자 ...



# Q&A



<https://github.com/PENEKhun>



<https://www.linkedin.com/in/penekhun>



[penekhun@gmail.com](mailto:penekhun@gmail.com)