# **API5:2019 مجوزدهی نادرست در سطح توابع**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  | |  | |
| **خاص API** | **قابلیت بهره‌برداری: 2** | **میزان شیوع: 2** | **قابلیت تشخیص: 1** | **پیامد فنی: 2** | **خاص کسب‌و‌کار** |
| بهره برداری از این ‌‌آسیب‌پذیری یعنی ارسال فراخوانی‌های API درست[[1]](#footnote-1) توسط مهاجم به سوی API Endpoint در ارتباط با فراخوانی‌هایی که مهاجم مجوز آنها را ندارد. این Endpointها ممکن است در معرض دید کاربران ناشناس، بدون مجوز یا عادی قرار داشته باشند. برای مهاجم تشخیص وجود چنین نواقصی در API آسان تر است چرا که ساختارمندتر بوده و نحوه دسترسی آنها به توابع، قابل پیش بینی تر است (مثلا تغییر متد HTTP از GET به PUT یا تغییر رشته “users” در URL به “admins”). | | کنترل‌های مجوزدهی برای توابع یا منابع غالبا در سطح پیکربندی یا کد مدیریت می شوند. بکارگیری کنترل‌های مناسب می‌تواند گیج کننده باشد چرا که اپلیکیشن‌های مدرن امروزی غالبا دارای انواع مختلفی از نقش‌ها و گروه‌ها و سلسله مراتب کاربری هستند (مثلا کاربران دارای بیش از یک نقش). | | چنین مشکلاتی منجر به دسترسی مهاجم به توابع غیرمجاز می‌شود. در این صورت توابع مدیریتی[[2]](#footnote-2) از جمله اهداف کلیدی مهاجم خواهند بود. | |

**آیا API از نظر مجوزدهی نادرست در سطح توابع[[3]](#footnote-3) ‌‌آسیب‌پذیر است؟**

بهترین راه یافتن مشکلات مجوزدهی در سطح توابع، تحلیل عمیق مکانیزم مجوزدهی با لحاظ کردن سلسله مراتب کاربران، نقش‌‌‌ها و گروهاه‌‌‌های متفاوت موجود در اپلیکیشن و پرسیدن پرسش‌‌‌های زیر است:

* آیا کاربر عادی می‌تواند به توابع و نقاط مدیریتی در API دسترسی داشته باشد؟
* آیا کاربری می‌تواند عمل حساسی که مجوز انجام آن را ندارد (نظیر ایجاد، تغییر یا حذف) را صرفا با تغییر متد HTTP (مثلا از GET به DELETE) انجام دهد؟
* آیا کاربری از گروه X می‌تواند صرفا با حدس زدن URLهای توابع و پارامترهای آن به مسیری (نظیر /api/v1/users/export\_all) که فقط باید برای کاربران گروه Y قابل مشاهده باشد دسترسی یابد؟

بایستی در نظر داشت که عادی یا مدیریتی بودن یک تابع در API (همان API Endpoint) صرفا بر مبنای مسیر URL تعیین نمی‌شود.

در حالیکه توسعه دهندگان بیشتر تمایل دارند که توابع مدیریتی را ذیل یک مسیر نسبی[[4]](#footnote-4) معین مانند api/admin قرار دهند، اما بسیار دیده می شود که این توابع مدیریتی در کنار توابع عادی در مسیرهایی نظیر api/users قرار داده شده‌اند.

**API5:2019 مجوزدهی نادرست در سطح توابع**

**مثال‌‌‌هایی از سناریوهای حمله**

**سناریو #1**

در خلال فرایند ثبت نام در یک اپلیکیشن که فقط به کاربران دعوت شده اجازه عضویت می‌دهد، اپلیکیشن موبایل، یک فراخوانی API به GET /api/invites/{invite\_guid} می‌فرستد. پاسخ دریافتی فایل JSONی را دارا است که درون آن اطلاعات دعوتنامه‌‌‌ها شامل نقش کاربر و آدرس ایمیل وی دیده می‌شود.

مهاجم درخواست مذبور را ضبط کرده و متد HTTP را به POST /api/invites/new تغییر می‌دهد. این تابع تنها بایستی از طریق کنسول مدیریت و برای ادمین‌‌‌ها قابل دسترسی باشد که بعلت عدم بکارگیری کنترل‌‌‌های صحیح مجوزدهی درسطح توابع اینگونه نیست.

در گام بعد مهاجم از این مساله بهره برداری کرده و برای خود دعوتنامه‌ای جهت ساخت یک اکانت ادمین می‌فرستد:

POST /api/invites/new

{“email”:”hugo@malicious.com”,”role”:”admin”}

**سناریو #2**

یک API دارای تابعی است که فقط ادمین‌‌‌ها بایستی آن را ببینند:

GET /api/admin/v1/users/all

این تابع در پاسخ جزئیات تمامی کاربران اپلیکیشن را برگردانده و کنترل‌‌‌های مجوزدهی در سطح توابع را نیز به درستی ‌‌‌‌پیاده‌سازی نکرده است. مهاجمی که با ساختار API آشنایی پیدا کرده، این مسیر را حدس زده و اطلاعات حساس تمامی کاربران اپلیکیشن را می‌رباید.

**چگونه از ‌‌آسیب‌پذیری مجوزدهی نادرست در سطح توابع پیشگیری کنیم؟**

ماژول مجوزدهی اپلیکیشن بایستی بطور یکپارچه توسط تمامی توابع اپلیکیشن فراخوانی شده و تحلیل آن نیز آسان باشد. همچنین در بیشتر مواقع، این روش حفاطتی توسط یک یا چند مولفه بیرونی و خارج از کد اصلی اپلیکیشن فراهم می‌شود.

* مکانیزم (های) اعمال شده بایستی بطور پیشفرض کلیه دسترسی‌‌‌ها را Deny (رد) نموده و برای دسترسی به هر یک از توابع، مجوزخاص دسترسی نقش مربوطه را طلب نمایند.
* توابع API از منظر عیوب مجوزدهی در سطح تابع با درنظر گرفتن منطق اپلیکیشن و سلسله مراتب گروه‌‌‌های کاربری مورد بازبینی قرار گیرد.

**API5:2019 مجوزدهی نادرست در سطح توابع**

* تمامی کنترلگرهای مدیریتی از یک کنترلگر مدیریتی انتزاعی که مجوزها را بر حسب نقش کاربر یا گروه پیاده‌سازی نموده، ارث بری داشته باشند.
* تمامی توابع مدیریتی درون یک کنترلگر عادی (غیرمدیریتی)، کنترل‌‌‌های مجوز مبتنی بر نقش کاربر یا گروه را بکارگیرند.

**مراجع**

**OWASP**

* [OWASP Article on Forced Browsing](https://owasp.org/www-community/attacks/Forced_browsing)
* [OWASP Top 10 2013-A7-Missing Function Level Access Control](https://owasp.org/www-project-top-ten/)
* [OWASP Development Guide: Chapter on Authorization](https://wiki.owasp.org/index.php/Category:Access_Control)

**خارجی**

* [CWE-285: Improper Authorization](https://cwe.mitre.org/data/definitions/285.html)

1. Legitimate [↑](#footnote-ref-1)
2. Administrative Functions [↑](#footnote-ref-2)
3. Broken Function Level Authentication [↑](#footnote-ref-3)
4. Relative Path [↑](#footnote-ref-4)