# **API2:2019 احرازهویت نادرست کاربر**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  | |  | |
| **خاص API** | **قابلیت بهره‌برداری: 3** | **میزان شیوع: 2** | **قابلیت تشخیص: 2** | **پیامد فنی: 3** | **خاص کسب‌و‌کار** |
| احرازهویت در APIها مکانیزمی پیچیده و سردرگم کننده است. در نتیجه امکان دارد مهندسین نرم افزار و امنیت تصورات غلطی درباره حد و مرز احرازهویت و نحوه ‌پیاده‌سازی آن داشته باشند. بعلاوه، مکانیزم احرازهویت هدفی بدیهی و آسان برای مهاجمان خواهد بود چرا که در معرض دید عموم قرار دارد. این دو نکته، مولفه احراز هویت را درمقابل بهره برداری‌ها و اکسپلویت‌های متعدد آسیب‌پذیر می‌سازد. | | در اینجا دو مساله وجود دارد:  1. نبود مکانیزم‌های حفاظتی: رفتار با نقاط و توابع مسئول احراز هویت در API بایستی متفاوت از سایر نقاط و توابع بوده و لایه‌های حفاظتی بیشتری داشته باشد.  2. ‌پیاده‌سازی نادرست مکانیزم حفاظتی: مکانیزم حفاظتی بدون لحاظ کردن بردارهای حمله[[1]](#footnote-1) یا با موارداستفاده[[2]](#footnote-2) نادرست (مثلا بکارگیری مکانیزم احرازهویتی که برای IoT طراحی شده در وب اپلیکیشن‌ها) ‌پیاده‌سازی یا استفاده شده‌اند. | | مهاجمین می‌توانند به حساب‌های کاربری سایر کاربران دسترسی یافته، اطلاعات شخصی آنها را خوانده و عملیات حساس (نظیر نقل و انتقالات مالی و ارسال پیام‌های شخصی) را از طرف آنها انجام دهد. | |

**آیا API از نظر احرازهویت نادرست کاربر[[3]](#footnote-3) آسیب‌پذیر است؟**

نقاط، توابع و جریان‌های احرازهویت API دارایی‌هایی هستند که بایستی محافظت شوند. همچنین توابع «فراموشی گذرواژه یا بازیابی گذرواژه» نیز بایستی در زمره مکانیزم‌های احرازهویت در نظر گرفته شوند.

یک API از منظر احرازهویت نادرست کاربر آسیب‌پذیر است اگر:

* اجازه حمله [درج هویت](https://www.owasp.org/index.php/Credential_stuffing)[[4]](#footnote-4) را بدهد که در آن مهاجم از لیستی از نام‌های کاربری و گذرواژه‌های معتبر استفاده می‌نماید.
* بدون استفاده از مکانیزم‌های CAPTCHA یا قفل کردن حساب کاربری[[5]](#footnote-5) اجازه حمله Brute Force روی یک حساب کاربری را بدهد.
* اجازه استفاده از گذرواژه‌های ضعیف را بدهد.
* جزئیات و داده‌های حساس مرتبط با احرازهویت از قبیل توکن‌های اصالت سنجی و گذرواژه‌ها را از طریق URL ارسال نماید.
* اصالت توکن‌ها را به بوته آزمون نگذارد.
* توکن‌ها JWT ضعیف یا بدون امضا (“alg”:”none”) را بپذیرد یا تاریخ انقضای آنها را اعتبارسنجی ننماید.
* از گذرواژه‌های آشکار[[6]](#footnote-6)، رمزگذاری نشده یا درهم سازی شده بصورت ضعیف[[7]](#footnote-7) استفاده نماید.
* از کلیدهای رمزگذاری ضعیف بهره ببرد.

**API2:2019 احرازهویت نادرست کاربر**

**مثال‌هایی از سناریوهای حمله**

**سناریو #1**

[درج هویت](https://www.owasp.org/index.php/Credential_stuffing) (استفاده از [لیستی از نام‌های کاربری یا گذرواژه‌های شناخته شده](https://github.com/danielmiessler/SecLists)) حمله‌ای رایج است. اگر اپلیکیشن از مکانیزم‌های حفاظتی خودکار در مقابل تهدیداتی نظیر درج هویت بهره نبرده باشد، آنگاه اپلیکیشن می‌تواند بعنوان یک پیشگوی گذرواژه[[8]](#footnote-8) یا آزمونگر جهت بررسی صحت اطلاعات هویتی جهت عبور از مکانیزم احرازهویت بکار رود.

**سناریو #2**

مهاجم جریان بازیابی گذرواژه را با ارسال یک درخواست POST به /api/system/verification-codes و ارائه نام کاربری در بدنه پیام آغاز می‌کند. سپس یک توکن پیامک 6 رقمی به تلفن قربانی ارسال می‌گردد. از آنجا که API خط مشی محدودیت سازی نرخ ارسال درخواست را بکار نگرفته، مهاجم می‌تواند تمامی جایگشت‌ها و ترکیبات محتمل را با استفاده از یک اسکریپت چندنخی[[9]](#footnote-9) با تابع زیر برای یافتن توکن صحیح ظرف چند دقیقه بیازماید.

/api/system/verification-codes/{smsToken}

**چگونه از ‌آسیب‌پذیری احرازهویت نادرست کاربر پیشگیری کنیم؟**

* حصول اطمینان از آنکه تمامی جریان‌های ممکن برای احراز هویت API (موبایل یا وب، سایر لینک‌هایی که از مکانیزم احرازهویت با یک کلیک و غیره) شناسایی شده است.
* مشورت با توسعه دهندگان و مهندسین در ارتباط با جریان‌های احرازهویتی که ممکن است از نظر دور مانده باشند.
* مطالعه و فهم کامل مکانیزم‌های احرازهویت استفاده شده در اپلیکیشن؛ بایستی درنظر داشت که OAuth و کلیدهای API نمی‌توانند بعنوان مکانیزمی برای احرازهویت به شمار آیند.
* در مساله احرازهویت، تولید توکن و ذخیره‌سازی گذرواژه، نباید چرخ را از ابتدا اختراع کرد بلکه بایستی از استانداردها استفاده نمود.
* توابع بازیابییا فراموشی گذرواژه بایستی از منظر محافظت در مقابل Brute Force، محدودسازی نرخ و قفل شدن حساب کاربری هم ارز با توابع و نقاط ورود[[10]](#footnote-10) در نظر گرفته شود.
* از راهنمای [احرازهویت OWASP](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html) استفاده شود.
* بکارگیری احرازهویت چندعاملی[[11]](#footnote-11)، در هر جا که امکان داشت.

**API2:2019 احرازهویت نادرست کاربر**

* بکارگیری مکانیزم‌های ضد Brute Force برای جلوگیری از حملات درج هویت، Dictionary و Brute Force بر روی توابع و نقاط احرازهویت در API. این مکانیزم بایستی سختگیرانه‌تر از مکانیزم محدودسازی نرخ معمول ‌پیاده‌سازی شود.
* بکارگیری مکانیزم‌های [قفل کردن حساب کاربری](https://www.owasp.org/index.php/Testing_for_Weak_lock_out_mechanism_(OTG-AUTHN-003)) / ‌CAPTCHA برای جلوگیری از حمله Brute Force علیه کاربران خاص.
* بکارگیری کنترل‌های مقابله با گذرواژه‌های ضعیف.
* کلیدهای API نبایستی برای احرازهویت کاربران بکار برده شود اما در عوض می‌تواند برای احرازهویت [اپلیکیشن/پروژه کلاینت](https://cloud.google.com/endpoints/docs/openapi/when-why-api-key) استفاده گردد.

**مراجع**

**OWASP**

* [OWASP Key Management Cheat Sheet](https://www.owasp.org/index.php/Key_Management_Cheat_Sheet)
* [OWASP Authentication Cheatsheet](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html)
* [Credential Stuffing](https://www.owasp.org/index.php/Credential_stuffing)

**خارجی**

* [CWE-798: Use of Hard-coded Credentials](https://cwe.mitre.org/data/definitions/798.html)

1. Attack Vectors [↑](#footnote-ref-1)
2. Use Case [↑](#footnote-ref-2)
3. Broken User Authentication [↑](#footnote-ref-3)
4. Credential Stuffing [↑](#footnote-ref-4)
5. Account Lockout [↑](#footnote-ref-5)
6. Plaintext [↑](#footnote-ref-6)
7. Weakly Hashed [↑](#footnote-ref-7)
8. Password Oracle [↑](#footnote-ref-8)
9. Multi-threaded [↑](#footnote-ref-9)
10. Login [↑](#footnote-ref-10)
11. Multi-factor Authentication [↑](#footnote-ref-11)