# **API4:2019 کمبود منابع و نبود محدودیت بر نرخ ارسال**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  | |  | |
| **خاص API** | **قابلیت بهره‌برداری: 2** | **میزان شیوع: 3** | **قابلیت تشخیص: 3** | **پیامد فنی: 2** | **خاص کسب‌و‌کار** |
| بهره برداری از این آسیب‌پذیری نیاز به ارسال درخواست‌های ساده‌ای به سوی API دارد و به احراز هویت هم نیازی نیست. کافی است تعدادی درخواست هم‌زمان از یک ماشین و یا با استفاده از منابع رایانش ابری به سوی API ارسال گردد تا بتوان از این آسیب‌پذیری بهره برد. | | یافتن APIهایی که محدودسازی نرخ ارسال را بکار نگرفته یا محدودیت‌های اعمال شده آنها ناکافی است، کار دشواری نیست. | | بهره برداری از این آسیب‌پذیری می‌تواند منجر به بروز DoS شده، در نتیجه API را از پاسخ به درخواست‌ها باز دارد و یا حتی آن را از دسترس خارج نماید. | |

**آیا API از نظر** **کمبود منابع و نبود محدودیت بر نرخ ارسال[[1]](#footnote-1) ‌‌آسیب‌پذیر است؟**

درخواست‌‌های ارسال شده به سوی API منابعی از قبیل پهنای باند شبکه، پردازنده، حافظه و فضای ذخیره‌سازی را مصرف می‌کنند. مقدار منابعی که برای پاسخگویی به یک درخواست صرف می‌شود عمدتا به ورودی‌‌های کاربر و منطق تجاری[[2]](#footnote-2) توابع API بستگی دارد. همچنین باید این موضوع را نیز درنظر داشت که درخواست‌‌های کلاینت‌‌های API مختلف برای دریافت منابع رقابت می‌کنند.

اگر دست‌کم یکی از محدودیت‌‌های زیر در سمت API به کلی اعمال نشده یا بطور نادرست (مثلا بیش از حد زیاد یا بیش از حد کم) ‌‌‌‌پیاده‌سازی شده باشد آنگاه API از منظر محدودیت یا کمبود نرخ ارسال، ‌‌آسیب‌پذیر خواهد بود:

* Time Out اجرا[[3]](#footnote-3)
* حداکثر میزان حافظه قابل تخصیص
* تعداد توصیف‌گر[[4]](#footnote-4) فایل‌‌ها
* تعداد پردازه‌‌ها
* اندازه محموله[[5]](#footnote-5) در درخواست‌‌ها (مثلا در هنگام آپلود)
* تعداد درخواست‌‌ها به ازای کلاینت یا منبع
* تعداد رکوردهایی که به ازای یک درخواست در یک صفحه نمایش داده می‌شوند.

**API4:2019 کمبود منابع و نبود محدودیت بر نرخ ارسال**

**مثال‌‌هایی از سناریوهای حمله**

**سناریو #1**

مهاجم از طریق ارسال یک درخواست POST به /api/v1/images اقدام به آپلود یک تصویر بزرگ می‌نماید. بعد از اتمام آپلود، API از روی تصویر آپلود شده تصاویرانگشتی[[6]](#footnote-6) متعددی با اندازه‌‌های مختلف ایجاد می نماید. به دلیل اندازه تصویر آپلودشده، حافظه‌ی دردسترس در خلال فرایند ایجاد تصاویر انگشتی تحت فشار قرار گرفته و API به وضعیت غیرپاسخگو[[7]](#footnote-7) می‌رسد.

**سناریو #2**

اپلیکیشنی لیست کاربران را در UI با محدودیت 200 کاربر در صفحه نمایش می‌دهد. لیست این کاربران از طربق ارسال پرس و جوی زیر از سرور دریافت می‌گردد: /api/users?page=1&size=100. در اینجا مهاجم می‌تواند با تغییر پارامتر size به 200000، مشکلاتی در عملکرد پایگاه داده پدید آورده و API را به وضعیت غیرپاسخگو برساند. در این حالت API قادر به پاسخگویی به هیچ درخواستی نخواهد بود (همان DoS).

همین سناریو را می‌توان به طریق مشابه برای ایجاد حملات سرریز Integer و سرریز Buffer استفاده نمود.

**چگونه از ‌‌آسیب‌پذیری کمبود منابع و نبود محدودیت بر نرخ ارسال پیشگیری کنیم؟**

* محدودسازی [حافظه](https://docs.docker.com/config/containers/resource_constraints/#memory)، [پردازنده](https://docs.docker.com/config/containers/resource_constraints/#cpu)، [تعداد دفعات راه اندازی مجدد](https://docs.docker.com/engine/reference/commandline/run/#restart-policies---restart)، [توصیف‌گرهای فایل](https://docs.docker.com/engine/reference/commandline/run/#set-ulimits-in-container---ulimit) و [پردازه‌‌ها](https://docs.docker.com/engine/reference/commandline/run/#set-ulimits-in-container---ulimit) با استفاده از Docker.
* اعمال محدودیت بر تعداد دفعاتی که در یک زمان مشخص امکان فراخوانی API وجود دارد.
* پس از ردشدن کلاینت از آستانه مجاز، این موضوع به همراه زمان رفع محدودیت به کلاینت اطلاع داده شود.
* افزودن اعتبارسنجی سمت سرور برای بررسی پارامترهای موجود در بدنه درخواست‌‌ها و رشته‌‌های پرس و جو، خصوصا مواردی که به نحوی با تعداد رکوردهای نمایش داده شده در پاسخ ارتباط دارند.
* تعریف و اِعمال بیشینه اندازه داده (نظیر بیشینه طول برای رشته‌‌ها یا بیشینه تعداد عناصر در آرایه‌‌ها) در درخواست‌‌ها و محموله‌‌های ورودی.

**API4:2019 کمبود منابع و نبود محدودیت بر نرخ ارسال**

**مراجع**

**OWASP**

* [Blocking Brute Force Attacks](https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks)
* [Docker Cheat Sheet - Limit resources (memory, CPU, file descriptors, processes, restarts)](https://github.com/OWASP/CheatSheetSeries/blob/3a8134d792528a775142471b1cb14433b4fda3fb/cheatsheets/Docker_Security_Cheat_Sheet.md#rule-7---limit-resources-memory-cpu-file-descriptors-processes-restarts)
* [REST Assessment Cheat Sheet](https://github.com/OWASP/CheatSheetSeries/blob/3a8134d792528a775142471b1cb14433b4fda3fb/cheatsheets/REST_Assessment_Cheat_Sheet.md)

**خارجی**

* [CWE-307: Improper Restriction of Excessive Authentication Attempts](https://cwe.mitre.org/data/definitions/307.html)
* [CWE-770: Allocation of Resources Without Limits or Throttling](https://cwe.mitre.org/data/definitions/770.html)
* “*Rate Limiting (Throttling)*” - [Security Strategies for Microservices-based Application Systems](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-204-draft.pdf), NIST

1. Lack of Resources and Rate Limiting [↑](#footnote-ref-1)
2. Business Logic [↑](#footnote-ref-2)
3. Execution Timeout [↑](#footnote-ref-3)
4. Descriptor [↑](#footnote-ref-4)
5. Payload [↑](#footnote-ref-5)
6. Thumbnail [↑](#footnote-ref-6)
7. Unresponsive [↑](#footnote-ref-7)