# **API3:2019 افشای مفرط داده**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  | |  | |
| **خاص API** | **قابلیت بهره‌برداری: 3** | **میزان شیوع: 2** | **قابلیت تشخیص: 2** | **پیامد فنی: 2** | **خاص کسب‌و‌کار** |
| بهره برداری از این آسیب‌پذیری آسان بوده و غالبا با شنود ترافیک به منظور تحلیل پاسخ‌های API برای یافتن داده حساسی که نباید به کاربر بازگردانده شود امکان پذیر است. | | APIها برای فیلتر کردن داده به کلاینت‌ها اتکا می‌کنند. از آنجا که APIها به عنوان منابع داده استفاده می‌شوند، توسعه دهندگان گاها آنها را بدون توجه به حساسیت اطلاعاتی که افشا می‌شود بکار می‌گیرند. ابزارهای خودکار غالبا نمی‌توانند این ‌آسیب‌پذیری را کشف کنند چرا که تمایز دادن بین داده مجازی که توسط API بازگردانده می‌شود با داده حساسی که نباید توسط API بازگردانده شود بدون داشتن فهمی عمیق از اپلیکیشن امکان پذیر نیست. | | افشای مفرط و بیش از حد داده معمولا منجر به افشای اطلاعات حساس می‌شود. | |

**آیا API از نظر افشای مفرط داده[[1]](#footnote-1) ‌آسیب‌پذیر است؟**

طراحی API به گونه‌ای است که داده حساس را به کلاینت باز می‌گرداند. این داده غالبا پیش از ارائه و نمایش به کاربر در سمت کلاینت فیلتر می‌شود. در نتیجه مهاجم می‌تواند براحتی و با شنود ترافیک، این داده حساس را مشاهده نماید.

**مثال‌هایی از سناریوهای حمله**

**سناریو #1**

تیم توسعه موبایل از /api/articles/{articleId}/comments/{commentId} برای مشاهده و پردازش فراداده[[2]](#footnote-2) کامنت‌ها بهره می‌برد. با شنود ترافیک اپلیکیشن موبایل، مهاجم در می‌یابد که داده مرتبط با نویسنده کامنت نیز بازگردانده می‌شود. این موضوع به این دلیل است که ‌‌پیاده‌سازی API از یک متد عمومی toJSON() برای سریالیزه کردن شیء User بهره می‌برد که این شی حاوی داده حساس PII[[3]](#footnote-3) می باشد.

**سناریو #2**

یک سیستم نظارتی مبتنی بر IoT به مدیران خود اجازه ایجاد کاربرانی با سطوح مجوز مختلف می‌نماید. یکی از مدیران یک حساب کاربری برای یک نیروی حفاظت فیزیکی (نگهبان) جدید می‌سازد که بر مبنای آن تنها امکان دسترسی به ساختمان‌های مشخصی بایستی وجود داشته باشد. به محض استفاده نگهبان مذبور از اپلیکیشن موبایل خود، یک فراخوانی API به سوی /api/sites/111/cameras روانه می‌شود تا اطلاعات مرتبط با دوربین‌های موجود را دریافت نموده و آنها را در دشبورد خود نمایش دهد. پاسخ، لیستی از جزئیات دوربین‌ها با فرمت زیر را در بردارد.

{“id”:”xxx”,”live\_access\_token”:”xxxx-bbbbb”,”building\_id”:”yyy”}

**API3:2019 افشای مفرط داده**

در حالیکه رابط گرافیکی کلاینت فقط دوربین‌هایی که نگهبان مذبور بایستی به آنها دسترسی داشته باشد را نشان می‌دهد، اما لیست کامل این دوربین‌ها در پاسخ API وجود دارد.

**چگونه از ‌آسیب‌پذیری افشای مفرط داده پیشگیری کنیم؟**

* عدم تکیه بر کلاینت در مساله فیلتر کردن داده حساس.
* بازبینی پاسخ دریافتی از API به منظور حصول اطمینان از آنکه فقط داده لازم و اصلی در آن نمایش داده می شود.
* پیش از افشا و در معرض دید عموم قراردادن یک API، مهندسین توسعه دهندگان Back-End بایستی از خود بپرسند: مصرف کننده و مخاطب این داده چه کسی است؟
* اجتناب از استفاده از متدهای عمومی to\_json() و to\_string() و در عوض دستچین کردن تک تک ویژگی‌ها و مشخصه‌هایی که برای پاسخ ضروری هستند.
* طبقه بندی اطلاعات حساس و شخصی[[4]](#footnote-4) ذخیره شده در APIها و بازبینی تمامی فراخوانی‌های APIهایی که این اطلاعات را باز می‌گردانند به منظور کشف و شناسایی مواردی که ضعف امنیتی در پی دارند.
* بکارگیری یک مکانیزم اعتبارسنجی الگومحور برای بررسی اعتبار پاسخ‌ها بعنوان یک لایه امنیتی دیگر و همچنین تعریف و اعمال این مکانیزم بر روی داده بازگردانده شده تمامی APIها از جمله خطاها.

**مراجع**

**خارجی**

* [CWE-213: Intentional Information Exposure](https://cwe.mitre.org/data/definitions/213.html)

1. Excessive Data Exposure [↑](#footnote-ref-1)
2. Metadata [↑](#footnote-ref-2)
3. Personally Identifiable Information [↑](#footnote-ref-3)
4. Personally Identifiable Information [↑](#footnote-ref-4)