# **API7:2019 پیکربندی امنیتی نادرست**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  | |  | |
| **خاص API** | **قابلیت بهره‌برداری: 3** | **میزان شیوع: 3** | **قابلیت تشخیص: 3** | **پیامد فنی: 2** | **خاص کسب‌و‌کار** |
| مهاجمین غالبا در تلاش برای یافتن حفره‌های وصله نشده، توابع رایج یا فایل‌ها و مسیرهای محافظت نشده به منظور دسترسی غیرمجاز به سیستم هستند. | | پیکربندی امنیتی نادرست می‌تواند در هر سطحی از API، از سطح شبکه تا سطح اپلیکشن روی دهد. ابزارهای خودکاری وجود دارند که فرایند تشخیص و بهره برداری از پیکربندی‌های نادرست نظیر تشخیص سرویس‌های غیرضروری را انجام می‌دهند. | | پیکربندی امنیتی نادرست نه تنها می‌تواند اطلاعات حساس کاربر را افشا کند بلکه جزئیاتی از سیستم که ممکن است به از دست رفتن کامل سرور منجر شود را نیز در معرض خطر قرار می‌دهد. | |

**آیا API از نظر پیکربندی امنیتی نادرست[[1]](#footnote-1) ‌‌‌آسیب‌پذیر است؟**

API از منظر پیکربندی امنیتی نادرست ‌‌‌آسیب‌پذیر است اگر:

* ایمن سازی امنیتی مناسب[[2]](#footnote-2) در هر قسمت از پشته اپلیکیشن رعایت نشده یا اپلیکیشن مجوزهای با پیکربندی نادرست روی سرویس‌‌‌‌های ابری داشته باشد.
* جدیدترین وصله‌‌‌‌های امنیتی نصب نشده و سیستم‌‌‌‌ها کاملا بروز نباشند.
* ویژگی غیرضروری (نظیر افعال اضافی HTTP) فعال باشند.
* امنیت لایه انتقال (TLS) غیرفعال باشد.
* دستورات و الزامات امنیتی (نظیر [سرایندهای امنیتی](https://www.owasp.org/index.php/OWASP_Secure_Headers_Project)) به سوی کلاینت ارسال نشوند.
* خط مشی اشتراک متقابل منابع (CORS[[3]](#footnote-3)) وجود نداشته یا به درستی ‌پیاده‌سازی نشده باشد.
* پیام‌‌‌‌های خطا ردپای پشته[[4]](#footnote-4) یا اطلاعات حساس دیگر را افشا نمایند.

**مثال‌‌‌‌هایی از سناریوهای حمله**

**سناریو #1**

مهاجم فایل .bash\_history را (که دستورات مورد استفاده تیم DevOps برای دسترسی به API را در خود دارد) در مسیر root سرور می‌یابد:

$ curl -X GET ‘https://api.server/endpoint/’ -H ‘authorization: Basic Zm9v0mJhcg==’

همچنین مهاجم خواهد توانست توابعی از API که تنها توسط تیم DevOps مورد استفاده قرارگرفته و مستند نشده‌اند را نیز بیابد.

**API7:2019 پیکربندی امنیتی نادرست**

**سناریو #2**

برای هدف قراردادن یک سرویس مشخص، مهاجم از موتورهای جستجوی رایج برای یافتن کامپیوترهایی که مستقیما توسط اینترنت قابل دسترسی هستند بهره می‌برد. در نتیجه، مهاجم میزبانی را می‌یابد که از یک سیستم مدیریت پایگاه داده محبوب استفاده نموده و اقدام به شنود روی پورت پیشفرض آن dbms می‌کند. از آنجا که میزبان از پیکربندی پیشفرض (که احراز هویت را بطور پیشفرض غیرفعال نموده) استفاده کرده، مهاجم می‌تواند به میلیون‌‌‌‌ها رکورد حاوی PII، داده‌‌‌‌های احرازهویت و ... دست یابد.

**سناریو #3**

مهاجم با بررسی ترافیک یک اپلیکیشن موبایل متوجه می‌شود که تمامی ترافیک HTTP بر بستر یک پروتکل ایمن (نظیر TLS) منتقل نمی‌شود و این موضوع خصوصا در زمان دانلود تصاویر پروفایل صدق می‌کند. از آنجا که تعامل کاربر با اپلیکیشن دودویی[[5]](#footnote-5) است، علیرغم انتقال ترافیک API بر بستر پروتکلی ایمن، مهاجم خواهد توانست الگویی را در اندازه پاسخ API شناسایی نموده و از آن برای رهگیری ترجیحات کاربر[[6]](#footnote-6) در خصوص محتوای ارائه شده به اپلیکیشن (مثلا تصاویر پروفایل) بهره ببرد.

**چگونه از ‌‌‌آسیب‌پذیری پیکربندی امنیتی نادرست پیشگیری کنیم؟**

چرخه حیات API بایستی شامل موارد زیر باشد:

* فرایندی تکرار شونده برای ایمن سازی API که منجر به ‌پیاده‌سازی سریع و آسان یک محیط ایمن شود.
* فرایندی برای بازبینی و بروزرسانی پیکربندی‌‌‌‌ها در سراسر پشته API؛ این بازبینی بایستی موارد از جمله بازبینی هماهنگی بین فایل‌‌‌‌ها، مولفه‌‌‌‌های API و سرویس‌‌‌‌های ابری (نظیر مجوزهای باکت‌‌‌‌های S3) را دربرگیرد.
* برقراری یک کانال ارتباطی ایمن برای دسترسی تمامی تعاملات API به دارایی‌‌‌‌های ایستا (نظیر تصاویر).
* فرایندی خودکار جهت ارزیابی پیوسته و مداوم اثربخشی پیکربندی و تنظیمات اعمال شده در سراسر محیط API و اپلیکیشن.

بعلاوه:

* برای جلوگیری از ارسال رهگیری رویدادهای استثنا و سایر داده‌‌‌‌های ارزشمند به مهاجم، در صورت امکان برای تمامی پاسخ‌‌‌‌های API (از جمله خطاها) الگوهای محموله[[7]](#footnote-7) مشخص تعریف و اعمال گردد.
* حصول اطمینان از اینکه API فقط به افعال HTTP مدنظر توسعه دهنده پاسخ می دهد و غیرفعال کردن سایر افعال (نظیر HEAD).

**API7:2019 پیکربندی امنیتی نادرست**

* APIهایی که انتظار می‌رود دسترسی به آنها از طریق کلاینت‌‌‌‌های مبتنی بر مرورگر (مثلا فرانت WebApp) باشد، بایستی خط مشی CORS مناسب را بکار گیرند.

**مراجع**

**OWASP**

* [OWASP Secure Headers Project](https://www.owasp.org/index.php/OWASP_Secure_Headers_Project)
* [OWASP Testing Guide: Configuration Management](https://www.owasp.org/index.php/Testing_for_configuration_management)
* [OWASP Testing Guide: Testing for Error Codes](https://www.owasp.org/index.php/Testing_for_Error_Code_(OTG-ERR-001))
* [OWASP Testing Guide: Test Cross Origin Resource Sharing](https://www.owasp.org/index.php/Test_Cross_Origin_Resource_Sharing_(OTG-CLIENT-007))

**خارجی**

* [CWE-2: Environmental Security Flaws](https://cwe.mitre.org/data/definitions/2.html)
* [CWE-16: Configuration](https://cwe.mitre.org/data/definitions/16.html)
* [CWE-388: Error Handling](https://cwe.mitre.org/data/definitions/388.html)
* [Guide to General Server Security](https://csrc.nist.gov/publications/detail/sp/800-123/final) , NIST
* [Let’s Encrypt: a free, automated, and open Certificate Authority](https://letsencrypt.org/)

1. Security Misconfiguration [↑](#footnote-ref-1)
2. Hardening [↑](#footnote-ref-2)
3. Cross-Origin Resource Sharing [↑](#footnote-ref-3)
4. Stack Trace [↑](#footnote-ref-4)
5. Binary [↑](#footnote-ref-5)
6. User Preferences [↑](#footnote-ref-6)
7. Payload Schema [↑](#footnote-ref-7)