# **API8:2019 تزریق ورودی‌های مخرب**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  | |  | |
| **خاص API** | **قابلیت بهره‌برداری: 3** | **میزان شیوع: 2** | **قابلیت تشخیص: 3** | **پیامد فنی: 3** | **خاص کسب‌و‌کار** |
| مهاجمین تلاش می‌کنند تا هرچه مسیر‌ برای تزریق (از جمله، ورودی‌های مستقیم، متغییرها و سرویس‌های یکپارچه) وجود دارد را با داده‌های مخرب پر کنند و انتظار دارند این اطلاعات بدست لایه مفسر برسد. | | وجود ‌آسیب‌پذیری تزریق ورودی‌های مخرب، امری متداول بوده و معمولا در پرس و جو‌های SQL، LDAP یا NoSQL، دستورات سیستم عامل، تجزیه کنندگان [[1]](#footnote-1)XML و ORM یافت می‌شود. این نقص‌ها به سادگی در زمان بازبینی کد منبع قابل کشف می‌باشند. مهاجمین نیز برای این منظور از اسکنرها و Fuzzerها استفاده می‌کنند. | | ‌آسیب‌پذیری تزریق ورودی‌های مخرب می‌تواند منجر به افشای اطلاعات و یا از دست رفتن اطلاعات شود. همچنین ممکن است این ضعف منجر به اختلال در سرویس‌دهی شده و یا حتی باعث از دست رفتن کامل دسترسی میزبان شود. | |

**آیا API از نظر نقص تزریق[[2]](#footnote-2) ورودی‌های مخرب آسیب‌پذیر است؟**

API از منظر نقصِ تزریق ورودی‌های مخرب ‌آسیب‌پذیر است اگر:

* API اطلاعات وارد شده توسط کاربر را اعتبار سنجی، فیلتر یا پاکسازی نکند.
* اطلاعات وارد شده توسط کاربر به صورت مستقیم استفاده شده و یا به انواع دستورات پرس و جو (SQL یا NoSQL یا LDAP) یا دستورات سیستم عامل، تجزیه‌کنندگان XML، ORM[[3]](#footnote-3) یا ORM[[4]](#footnote-4) افزوده شود.
* API اطلاعات دریافت شده از سیستم‌های خارجی را اعتبار سنجی، فیلتر یا پاکسازی نکند.

**مثال‌هایی از سناریوهای حمله**

**سناریو #1**

میان‌افزار یک دستگاه کنترل (فرزندان) توسط والدین تابعی را در مسیر /api/CONFIG/restore ارائه می‌کند، که انتظار دارد مقدار appId ارسال شده به سمت آن، دارای مقادیر چند متغیره باشد. با استفاده از یک دیکامپایلر، مهاجم در می‌یابد مقدار appId بدون هیچ‌گونه تغییر یا پاکسازی‌ به یک فراخوانی سیستمی[[5]](#footnote-5) ارسال می‌شود.

snprintf(cmd, 128, "%srestore\_backup.sh /tmp/postfile.bin %s %d",  
 "/mnt/shares/usr/bin/scripts/", appid, 66);  
system(cmd);

دستور زیر به مهاجم این امکان را می‌دهد تا هر دستگاهی با این میان افزار ‌آسیب‌پذیر را خاموش کند.

$ curl -k "https://${deviceIP}:4567/api/CONFIG/restore" -F  
'appid=$(/etc/pod/power\_down.sh)'

**API8:2019 تزریق ورودی‌های مخرب**

**سناریو #2**

سامانه تحت وب ساده‌ای با عملکرد‌های اولیه CRUD[[6]](#footnote-6)، برای انجام عملیات‌های رزرو وجود دارد. مهاجم موفق به شناسایی تزریق NoSQL از طریق متغیر bookingId در رشته پرس و جو و در درخواست حذف رزرو شده است. درخواست مذکور شبیه به DELETE /api/booking?bookingId=678 می‌باشد.

سرور API از تابع زیر برای رسیدگی کردن به درخواست‌های حذف استفاده می‌کند:

router.delete('/bookings', async function (req, res, next) {  
 try {  
 const deletedBooking = await Bookings.findOneAndRemove({\_id' :req.query.bookingId});

res.status(200);  
} catch (err) {  
 res.status(400).json({  
 error: 'Unexpected error occured while processing a request'  
 });  
}  
});

مانند آنچه در زیر مشاهده می‌کنید، مهاجم پس از رهگیری درخواست و تغییر مقدار bookingId استفاده شده در رشته پرس و جو، می‌تواند رزرو انجام شده توسط کاربر دیگری را حذف نماید.

DELETE /api/bookings?bookingId[$ne]=678

**چگونه از آسیب‌پذیری تزریق ورودی‌های مخرب پیشگیری کنیم؟**

جلوگیری از ‌آسیب‌پذیری تزریق ورودی‌های مخرب، نیازمند جداسازی اطلاعات از دستورات و پرس و جو‌ها می‌باشد.

* داده‌ها باید توسط یک کتابخانه پایدار، فعال و قابل اطمینان اعتبار سنجی شود.
* تمامی اطلاعات وارد شده توسط کاربران و دیگر سیستم‌های یکپارچه باید اعتبار سنجی، فیلتر یا پاکسازی شود.
* کاراکترهای خاص باید توسط قوانین مشخص برای مفسران نهایی تغییر داده شود.
* همواره تعداد رکوردهای بازگردانده شده باید محدود شود تا در صورت وجود نقص تزریق ورودی‌های مخرب، از افشای انبوه اطلاعات جلوگیری شود.
* داده‌های ورودی باید توسط فیلترهای مناسب اعتبار سنجی شود تا تنها مقادیر معتبر برای هر پارامتر ورودی مجاز به وارد شدن باشند.
* برای تمامی متغییر‌های رشته‌ای، نوع داده و الگوی سخت‌گیرانه‌ای تعریف شود.

**API8:2019 تزریق ورودی‌های مخرب**

**مراجع**

**OWASP**

* [OWASP Injection Flaws](https://owasp.org/www-community/Injection_Flaws)
* [SQL Injection](https://owasp.org/www-community/attacks/SQL_Injection)
* [NoSQL Injection Fun with Objects and Arrays](https://owasp.org/www-pdf-archive/GOD16-NOSQL.pdf)
* [Command Injection](https://owasp.org/www-community/attacks/Command_Injection)

**خارجی**

* [CWE-77: Command Injection](cwe.mitre.org/data/definitions/77.html)
* [CWE-89: SQL Injection](https://cwe.mitre.org/data/definitions/89.html)

1. Parser [↑](#footnote-ref-1)
2. Injection [↑](#footnote-ref-2)
3. Object Relational Mapping [↑](#footnote-ref-3)
4. Object Document Mapper [↑](#footnote-ref-4)
5. System Call [↑](#footnote-ref-5)
6. Create, Read, Update, Delete [↑](#footnote-ref-6)