
بسم الله الرحمن الرحيم

آسیب‌پذیری‌های API

استاد: زینب آقایی

رسول کامکار – سعید قاسمی

ارائه پروژه: تیر ۱۴۰۱

فهرست

مقدمه

امروزه یک عنصر بنیادی نوآوری در این دنیای پر از برنامه‌های کاربردی که با سرعت نیز درحال پیشروی است، مفهوم Application Programming Interface یا API است که بخش حیاتی از بانک‌داری، فروشگاه‌ها، اینترنت اشیا، حمل‌ونقل و درکل خدمات تحت وب است.

این نوآوری به طور ذاتی منطق نرم‌افزار و اطلاعات حساس مانند Personally Identifiable Information (PII) را در معرض دید قرار می‌دهد و به همین دلیل تبدیل به هدفی با اولویت بالا برای حمله مهاجمان شده است. بنابراین شناخت آسیب‌پذیری‌های متداول APIها و تلاش و برای رفع و جلوگیری از آن‌ها به فراهم‌سازی بستری امن برای ارائه خدمات کمک قابل توجهی می‌کند.

به همین دلیل امنیت APIها و تحلیل آسیب‌پذیری‌های آن‌ها مورد توجه ویژه کارشناسان این حوزه قرار گرفته است. انجمن امنیتی OWASP (Open Web Application Security Project) یکی از فعالان این حوزه است که یکی از پروژه‌های آن‌ها بررسی ۱۰ آسیب‌پذیری بزرگ APIها است.

در ادامه این مقاله به بررسی تعدادی از مهم‌ترین و متداول‌ترین دسته‌بندی‌های آسیب‌پذیری‌ها که در این پروژه مورد بحث قرار گرفته‌اند پرداخته می‌شود.

بخش ۱- Broken Object Level Authorization

در این نوع آسیب‌پذیری، بخاطر عدم بررسی وضعیت و هویت کاربر توسط سرور و اینکه بررسی درخواست‌ها تنها وابسته به پارامتر هایی مانند ID اشیاء است، حمله‌کننده می تواند صرفا با تغییر ID اشیاء داخل درخواست‌هایی که به API می زند، به اطلاعات مهم دسترسی پیدا کند و آن‌ها را تغییر دهد یا حتی حذف کند.

این حمله یکی از تاثیرگذارترین و متداول‌ترین حملات بوده است. با اینکه برنامه‌ها زیرساخت‌هایی برای بررسی اجازه دسترسی فراهم می‌کنند، طراحان و توسعه‌دهندگان API ممکن است استفاده از اینگونه امکانات را فراموش کنند.

مثال

در یک API اطلاعات کاربران را می توان با الگوی /users/{userID}/uinfo.json به‌دست آورد. همچنین از طریق یکی دیگر از بخش‌های API می‌توان به لیست کاربران و IDهای آن‌ها دست یافت. حال حمله کننده می‌تواند به کمک یک اسکریپت IDها را در الگوی داده شده جای‌گذاری کرده و اطلاعات مربوط به تمام کاربران را به‌دست آورد.

جلوگیری

پیاده‌سازی بستر کنترل دسترسی مناسب که بر اساس نوع کاربر و نقش‌های داده شده (و نه فقط شیئ مورد نظر) به درخواست‌ها پاسخ می‌دهد. در این بستر بررسی می‌شود که آیا کاربر اجازه عمل درخواست شده را دارد یا نه.

پیاده‌سازی تست‌هایی برای ارزیابی مکانیزم‌های احراز مجوز.

بخش ۲- Broken User Authentication

این آسیب‌پذیری مربوط به پیکربندی ضعیف یا نامناسب بخش احراز هویت API می‌شود. گستردگی این نوع آسیب‌پذیری از پیچیدگی مکانیزم‌های احراز هویت ناشی می‌شود. همچنین به دلیل اینکه بخش احراز هویت API در دسترس عموم است، این بخش تبدیل به یک هدف آسان و مهم برای حمله‌کنندگان است. این بخش باید شامل تمهیدات محافظتی بیشتری نسبت به بقیه بخش‌ها داشته باشد و همچنین مکانیزم‌های کنترلی باید بر اساس کاربرد و نوع حملات ممکن تعیین و تنظیم شوند. عدم رعایت این‌ها زمینه‌ساز این آسیب‌پذیری است. در صورت موفقیت حملات، مهاجم به حساب دیگر کاربران و اطلاعات آن‌ها دسترسی پیدا می‌کند و می‌تواند از طرف آن‌ها کارهایی مانند تراکنش‌های مالی یا فرستادن پیام‌های شخصی انجام دهد.

چندی از مواردی که باعث آسیب‌پذیری API می‌شود:

- از credential stuffing جلوگیری نمی‌کند. در credential stuffing هنگام لو رفتن اطلاعات یک دیتابیس، اطلاعات مربوط به احراز هویت آن به طور گسترده و به کمک ابزارهایی روی تعداد زیادی از سرورها و APIها تست می‌شود تا در صورتی که تعدادی از کاربران نام کاربری و رمزعبور یکسان روی سرویس‌های مختلف دارند، حمله کننده بتواند روی آن سرویس‌ها نیز به اکانت کاربران دسترسی پیدا کند.
- اجازه حمله Brute force را به حمله‌کنندگان می‌دهد و از تمهیداتی مانند CAPTCHA استفاده نمی‌کند.
- اجازه انتخاب رمز عبورهای کوتاه و ضعیف را می‌دهد.
- اطلاعات حساس مربوط به احراز هویت را همراه URL می‌فرستد.
- از کلیدهای رمزنگاری ضعیف استفاده می‌کند.

مثال

یک ارائه‌دهنده VPS در پیکربندی سیستم‌های ویندوزی خود برای اتصال کلاینت‌ها از Remote desktop connection روی پورت پیش فرض 3389 استفاده می‌کند. از آنجا که به طور پیش‌فرض محدودیتی روی نرخ

درخواست‌های ورودی وجود ندارد، حمله‌کننده می‌تواند به Dictionary Attack (برای حدس رمزعبورهایی که احتمالا بر اساس نام‌کشور یا دیتاسنتر VPS تنظیم می‌شوند) اقدام کند.

جلوگیری

بررسی و تحلیل همه روش‌های احراز هویت

استفاده از استانداردها و ابزارهای آماده‌ی مورد تایید و اثبات شده برای احراز هویت

استفاده از احراز هویت چند مرحله‌ای در مواردی که امکان دارد

به کارگیری مکانیزم‌های جلوگیری از Brute force و Credential stuffing

وادار کردن کاربران به انتخاب رمز عبور قوی و مناسب

بخش ۳- Excessive Data Exposure

سواستفاده از این آسیب‌پذیری بسیار ساده است، کافیست شنود ترافیک و آنالیز پاسخ API برای یافتن اطلاعات حساس که نباید در اختیار کاربر قرار گیرد انجام شود. این اتفاق زمانی رخ می‌دهد که توسعه دهنده پیاده‌سازی را به طور کلی انجام دهد و API در پاسخ‌های خود تمامی اطلاعات مربوط به یک بخش را ارسال کند و سمت کلاینت وظیفه فیلتر کردن اطلاعات مورد نیاز برای نمایش به کاربر را به عهده داشته باشد.

شناسایی این آسیب‌پذیری توسط ابزارهای خودکار بسیار دشوار است زیرا نمی‌توان حساسیت اطلاعات فرستاده شده را بدون درک عمیقی از برنامه کاربری متوجه شد.

مثال

تیم طراحی برنامه اندروید از endpoint با آدرس
/api/movies/{movieID}/comments/{commentID} برای نمایش اطلاعات یک نظر مربوط به فیلم استفاده می‌کند، مهاجم با شنود شبکه متوجه می‌شود که اطلاعات حساس مربوط به نویسنده آن نظر نیز ارسال می‌شود.

جلوگیری

عدم اعتماد به سمت کلاینت برای فیلتر کردن اطلاعات حساس

بازبینی پاسخ API برای شناسایی اطلاعات حساس و اطمینان از اینکه تنها اطلاعات مورد نیاز فرستاده می‌شوند

مهندسین back-end قبل از بازکردن یک endpoint جدید باید درباره اینکه "چه کسی مصرف کننده اطلاعات است" سوال کنند

خودداری از آماده‌سازی کلی تمامی اطلاعات برای ارسال (مانند استفاده از تابع to_json) و آماده کردن حداقل اطلاعات که مورد نیاز هستند و ارسال آن‌ها

بخش ٤- Lack of Resources & Rate Limiting

چندین درخواست ساده API که حتی لزوماً نیاز به احراز هویت هم ندارند به صورت همزمان از یک یا چند منبع کامپیوتری ارسال شده که باعث مصرف بیش از حد منابع شبکه، پردازش، حافظه و ذخیره‌سازی و در نهایت به منع ارائه سرویس (DoS) ختم می‌شود. معمولا APIها محدودیتی بر روی نرخ درخواست‌ها ندارند یا محدودیت نامناسبی اعمال کرده‌اند. از جمله مواردی که عدم اعمال محدودیت بر آن‌ها می‌تواند بستر حمله باشد

- حافظه اختصاص یافته
- حجم یا تعداد درخواست‌ها
- تعداد پروسه‌ها
- تعداد فایل‌های باز شده

مثال

یک سرویس انتقال فایل در بستر پیاده‌سازی خود از برنامه‌نویسی Socket به صورت multi-process استفاده کرده و با برقراری هر اتصال جدید یک پروسه فرزند می‌سازد اما تعداد اتصالات برقرار شده ثبت و کنترل نمی‌شود. مهاجم با برقراری تعداد زیادی اتصال، به سرعت منابع موجود را مصرف کرده و به سادگی ارائه سرویس را مختل می‌کند.

جلوگیری

استفاده از ابزارهایی مانند Docker برای محدود کردن مصرف منابع

محدود کردن تعداد فراخواهی‌های API سمت کلاینت در یک بازه زمانی و همینطور اطلاع دادن به کاربر از زمان وقوع و اتمام این محدودیت

محدود کردن و اعمال حداکثر اندازه بر مولفه‌های ورودی API مانند تعداد عناصر آرایه بازگشتی یا بزرگترین طول رشته

بخش ۵- Broken Function Level Authorization

این نقض امنیتی به مهاجم امکان ارسال درخواست‌های درست و مجاز به endpointهایی که نباید به آن‌ها دسترسی داشته باشد را می‌دهد. کشف کردن این نوع نقض‌ها نسبتاً ساده است، مثلا جایگزین کردن users با admins در URL. به دلیل پیچیدگی برنامه‌های مدرن و ساختار کاربری آن‌ها، پیاده‌سازی بررسی‌های مجوز، کاری گیج‌کننده و پرخطا است که در نهایت منجر به دسترسی مهاجمان به نقاط غیرمجاز می‌شود.

مثال

در یک برنامه کاربردی تنها ادمین‌های سیستم دسترسی اضافه کردن کاربر را دارند که API مربوط به این عملیات در آدرس api/users/new/ قرار گرفته است. از طرفی، درخواست ورودی به سیستم کاربران ثبت شده به آدرس api/users/login/ ارسال می‌شود، مهاجم با حدس زدن URL ثبت کاربر جدید و عدم بررسی مجوز صحیح در endpoint ثبت کاربر، یک حساب کاربری جدید با مشخصات دلخواه به وجود می‌آورد.

جلوگیری

برنامه باید یک واحد بررسی مجوز داشته باشد که تمامی بخش‌های API به راحتی بتوانند از آن استفاده و مجوز درخواست‌ها را بررسی کنند

مکانیسم‌های اجرایی این واحد باید همه دسترسی‌ها را به صورت پیشفرض رد کند و تنها به نقش‌های مشخصی دسترسی‌های خاص به هر بخش را بدهد

بخش ٦- Mass Assignment

فریمورک‌های مدرن به توسعه‌دهنده‌گان پیشنهاد می‌کنند که از توابعی برای تبدیل خودکار ورودی به متغیرهای داخل کد استفاده کنند. به طور مثال یک کاربر نباید بتواند مقدار مولفه‌های مربوط به دسترسی مانند user.is_admin یا مولفه‌های وابسته به پردازش مانند user.balance را تغییر دهد، اما با تبدیل خودکار پارامترهای سمت کلاینت به پارامترهای اشیا داخلی برنامه این کار امکان پذیر است. این امر موجب می‌شود مهاجمین بتوانند اطلاعات حساسی از شیئ را تغییر دهند. این آسیب‌پذیری نیازمند درکی از منطق برنامه کاربردی، روابط بین اشیا و ساختار API دارد که می‌تواند منجر به بالا رفتن سطح دسترسی مهاجم، دستکاری اطلاعات، گذر از مکانیزم-های امنیتی و موارد دیگر شود. طراحی APIها به اینگونه است که ذاتا سطوح پایین پیاده‌سازی را افشا کرده که باعث سوءاستفاده راحت‌تر از این نوع آسیب‌پذیری می‌شود.

مثال

یک سرویس ذخیره‌سازی ابری به کاربران اجازه VIP شدن پس از پرداخت هزینه می‌دهد، هنگام درخواست دریافت مشخصات کاربر یک درخواست GET به api/v1/users/me/ توسط کلاینت ارسال می‌شود که مشخصات کاربر از جمله ویژه بودن آن را ارسال می‌کند.

{username”:”anon”,”age”:21,”is_vip”:false“}

همینطور کلاینت برای تغییر مشخصات معمولی کاربر یک درخواست POST به همین آدرس ارسال می‌کند.

{username”:”anon”,”age”:22“}

با ارسال یک درخواست POST به همراه پارامتر is_vip و وجود آسیب‌پذیری Mass assignment، مهاجم می‌تواند نوع کاربری خود را به ویژه تغییر دهد.

{username”:”anon”, ”age”:22, “is_vip”:true“}

جلوگیری

تا حد ممکن از ابزارهای تبدیل ورودی کلاینت به متغیرها و پارامترهای کد استفاده نشود

ویژگی‌هایی (property) که کلاینت مجاز به تغییر آن‌ها است، در لیست سفید و ویژگی‌های حساس در لیست سیاه قرار گیرند

در صورت امکان، ساختارهایی برای اطلاعات ورودی کاربران مشخص کرده و برروی پارامترهای درخواست‌ها اعمال شود

بخش ۷- Security Misconfiguration

مهاجمان برای به دست آوردن دسترسی غیرمجاز یا دانش باارزش از وضعیت و ساختار سیستم، به جست‌وجوی عیوب برطرف نشده، endpointهای متداول یا فایل‌ها و دایرکتوری‌های محافظت نشده می‌پردازند که با پیکربندی نادرست در هر سطحی از API stack، منجر به فاش شدن اطلاعات حساس کاربران یا جزییات سیستم و به خطر افتادن آن می‌شود. به روز نبودن سیستم، فعال بودن قابلیت‌های اضافه مانند HTTP verbs، واضح و دردسترس بودن پیام‌های خطا مانند stack trace یا اطلاعات حساس و مواردی از این قبیل نشان دهنده وجود این آسیب‌پذیری است.

مثال

یک سامانه اشتراک فایل، از سیستم‌های به روز نشده که پورت SMB (Server Message Block) آسیب‌پذیر دارند استفاده می‌کند، علیرغم منتشر شدن بروزرسانی‌ها برای سیستم عامل استفاده شده در این سامانه، بدافزارها از این آسیب‌پذیری استفاده کرده و سیستم‌های این سامانه را آلوده کرده‌اند.

جلوگیری

یک پروسه مستحکم‌سازی قابل تکرار که به آسانی و با سرعت گسترش یک فضای به خوبی ایزوله شده را مهیا می‌سازد

بازبینی و به‌روزرسانی پیکربندی‌ها در سرتاسر API stack

یک کانال امن ارتباطی برای تمامی دسترسی‌های API به منابع Static (مانند عکس‌ها)

یک پروسه خودکار برای ارزیابی مکرر پیکربندی و تنظیمات در سرتاسر فضای سیستم

غیرفعال کردن قابلیت‌های غیرضروری مانند stack trace و exception trace برای جلوگیری از فاش شدن این نوع اطلاعات

بخش ۸- Injection

API هرجا که ورودی‌هایی را از کاربر می گیرد، ممکن است نسبت به Injection آسیب‌پذیر باشد که به حمله-کننده امکان جایگذاری داده‌های دلخواه خود در بدنه و پارامترهای درخواست‌ها برای تزریق آنها از طریق API را می‌دهد به امید اینکه این داده‌ها به عنوان دستورات و مفاهیم خاص آن محیط، شناخته شده و تفسیر و اجرا شوند. این محیط‌ها می توانند در قالب کوئری‌های SQL و NoSQL یا دستورات سیستم‌عامل باشند. این حملات می‌توانند باعث نشت و تخریب اطلاعات، DoS و یا دسترسی کامل به سیستم مقصد شوند. تشخیص این آسیب پذیری با بررسی سورس کد و ابزارهای فرستادن ورودی های تصادفی و نیمه‌تصادفی آسان‌تر می‌شود. بستر این آسیب‌پذیری وقتی فراهم می‌شود که ورودی‌هایی که از سمت کاربر می‌آید به درستی بررسی و فیلتر نمی‌شود.

مثال

برنامه ای برای انجام نوعی پردازش نام یک فایل را به عنوان پارامتر گرفته و آن را بصورت یک رشته به دستورات شل مورد نیاز خود اضافه کرده و با استفاده از تابع system() آن‌ها را اجرا می‌کند. در این وضعیت حمله‌کننده می-تواند با اضافه کردن && و دستورات مورد نظر خود پس از آن (مانند "some_file && ls") دستوراتی را اجرا کند که در صورتی که برنامه بنا به کاربرد خود، دسترسی روت داشته باشد امکان اجرای هر دستوری را به مهاجم می‌دهد.

جلوگیری

استفاده از یک کتابخانه قابل اعتماد برای بررسی ورودی کاربر و اعمال آن روی تمام داده‌های ورودی

ترجیح بر ساختار پارامتری API

تعیین حد بالا برای تعداد رکوردهایی که در پاسخ فرستاده می‌شوند، برای کاهش تاثیر نشت اطلاعات در صورت حمله

تعریف ساختار دقیق برای رشته‌های ورودی

بخش ۹- Improper Assets Management

نسخه های قدیمی API معمولا معیوب هستند که به راحتی باعث در خطر قرار گرفتن سیستم می‌شوند، بدون اینکه نیاز باشد مهاجم با مکانیزم‌های امنیتی که برای حفاظت از نسخه‌های جدید تعبیه شده‌اند درگیر شود. استراتژی‌های قدیمی، نبود فهرست منابع (assets inventory) و نبود برنامه‌ای برای کنار گذاشتن نسخه‌های قدیمی به مهاجم امکان دسترسی به اطلاعات حساس یا حتی به دست آوردن کنترل کل سیستم را می‌دهد، زیرا یک نسخه قدیمی معیوب API به یک سیستم مجهز به مفاهیم مدرن مانند microserviceها متصل است. این مشکلات زمانی رخ میدهد که مستندات، فهرست‌های موجودی و فهرست سرویس‌های پیاده شده موجود نیست یا قدیمی هستند و یا نسخه-های قدیمی API در حال استفاده است و برنامه‌ای برای کنار گذاشتن آن‌ها در دسترس نیست.

مثال

- یک سرویس ایمیل پس از شناسایی یک آسیب‌پذیری حیاتی، پیاده‌سازی API خود را تغییر داده و مستندات را نیز به روز کرده اما همچنان endpointهای مربوط به نسخه آسیب‌پذیر در دسترس است. مهاجم پس از کاوش در سرویس‌های آرشیو اینترنت (مانند archive.org) به مستندات نسخه قدیمی دست پیدا کرده و پس از بررسی و اطمینان از فعال بودن سرویس‌ها، به اطلاعات میلیون‌ها کاربر دسترسی پیدا کرده است.
- یک وبسایت با استفاده از فریمورک Django بک-اند خود را پیاده‌سازی کرده است. توسعه‌دهندگان قبل از تحویل پروژه، فراموش کرده‌اند گزینه DEBUG را غیرفعال کنند. در این صورت هنگام وقوع خطا در اجرای کد یا اررورهای دسته 4xx پروتکل HTTP، جزییاتی از ساختار بک-اند و آسیب‌پذیری‌های احتمالی منتشر می‌شود که می‌تواند بسترسازی برای حملات باشد.

جلوگیری

فهرست و مستندسازی API host با تمرکز بر محیط API (تولید، تست و توسعه)، نحوه دسترسی به آن (عموم، داخلی، شرکا) و نسخه‌های API

فهرست و مستندسازی سرویس‌های استفاده شده با تمرکز بر نقش آن‌ها در سیستم، جریان اطلاعات و میزان حساسیت آن‌ها

مستندسازی تمامی جنبه‌های API مانند احراز اصالت، خطاها، rate limiting، CORS، endpointها و پارامترهای آنان

دردسترس قرار دادن مستندات API تنها به افراد مجاز به استفاده از آن

استفاده از اقدامات امنیتی برای تمامی نسخه‌های در معرض استفاده API و نه فقط نسخه کنونی

بخش ۱۰- Insufficient Logging & Monitoring

در صورت عدم یا ناکافی بودن گزارش‌گیری و نظارت در سیستم، مهاجمین می‌توانند به راحتی و بدون دیده شدن، از سیستم سواستفاده کنند و برای اینکار نیز زمان بسیار زیادی دارند. زمانی که API لاگی تولید نمی‌کند، سطح تولید لاگ یا محتوای پیام‌ها درست تنظیم نشده و یا این گزارش‌ها مرتبا نظارت و بررسی نمی‌شوند، می‌تواند به کاهش امنیت سیستم ختم شود. حتی در صورت تولید لاگ مناسب، اگر Integrity آن تضمین نشود، نمی‌توان نظارت درست و مطمئنی بر روی سیستم داشت. Log Injection یک نمونه حمله برای از بین بردن Integrity لاگ‌ها است که با تولید گزارشات غیر واقعی و هدفدار یا تخریب کامل فایل گزارش، می‌تواند رد حمله را بپوشاند یا حتی مهاجم را شخص دیگری جلوه دهد.

مثال

فایروال یک شبکه به دلیل برنامه‌ریزی با دقت پایین، پیام‌های اخطار False Positive به ادمین ارسال می‌کند. ادمین پس از بررسی لاگ‌های سیستم متوجه این موضوع شده اما اقدامی برای رفع اشکال فایروال انجام نمی‌دهد. پس از مدتی این شبکه تحت یک حمله گسترده Dictionary قرار می‌گیرد و مدیر سیستم به دلیل سابقه قبلی، پیام‌های اخطار را نادیده گرفته و در نتیجه عدم مقابله با این حمله، اطلاعات حساب چندین هزار کاربر افشا شده است.

جلوگیری

ثبت گزارش تمامی تلاش‌های احراز اصالت ناموفق، دسترسی‌های منع شده و خطاهای اعتبارسنجی ورودی‌ها

ذخیره گزارشات با یک قالب مناسب برای استفاده و همچنین با اطلاعات کافی برای شناسایی مهاجمین و خرابکاران

تضمین Integrity گزارشات و طبقه‌بندی آن‌ها به عنوان اطلاعات حساس

استفاده از ابزارهای نظارت برای بررسی مرتب و مداوم زیرساخت، شبکه و عملکرد API

منابع

CWE-1059: Insufficient Technical Documentation. (n.d.). Retrieved from https://cwe.mitre.org/data/definitions/1059.html

CWE-16: Configuration. (n.d.). Retrieved from https://cwe.mitre.org/data/definitions/16.html

CWE-213: Exposure of Sensitive Information Due to Incompatible Policies. (n.d.). Retrieved from https://cwe.mitre.org/data/definitions/213.html

CWE-223: Omission of Security-relevant Information. (n.d.). Retrieved from https://cwe.mitre.org/data/definitions/223.html

CWE-284: Improper Access Control. (n.d.). Retrieved from https://cwe.mitre.org/data/definitions/284.html

CWE-285: Improper Authorization. (n.d.). Retrieved from https://cwe.mitre.org/data/definitions/285.html

CWE-307: Improper Restriction of Excessive Authentication Attempts. (n.d.). Retrieved from https://cwe.mitre.org/data/definitions/307.html

CWE-639: Authorization Bypass Through User-Controlled Key. (n.d.). Retrieved from https://cwe.mitre.org/data/definitions/639.html

CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection'). (n.d.). Retrieved from https://cwe.mitre.org/data/definitions/77.html

CWE-770: Allocation of Resources Without Limits or Throttling. (n.d.). Retrieved from https://cwe.mitre.org/data/definitions/770.html

CWE-778: Insufficient Logging. (n.d.). Retrieved from https://cwe.mitre.org/data/definitions/778.html

CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection'). (n.d.). Retrieved from https://cwe.mitre.org/data/definitions/89.html

CWE-915: Improperly Controlled Modification of Dynamically-Determined Object Attributes. (n.d.). Retrieved from https://cwe.mitre.org/data/definitions/915.html

Forced browsing. (n.d.). Retrieved from OWASP: https://owasp.org/www-community/attacks/Forced_browsing

Mueller, N. (n.d.). *Credential stuffing*. Retrieved from OWASP : https://owasp.org/www-community/attacks/Credential_stuffing

OWASP API Security Project. (n.d.). Retrieved from https://owasp.org/www-project-api-security/