OWASP API Security Top 10 2019

Daftar Isi

Tentang OWASP

Open Web Application Security Project (OWASP) adalah komunitas terbuka yang didedikasikan untuk memungkinkan organisasi mengembangkan, membeli, dan memelihara aplikasi dan API yang dapat dipercaya.

Di OWASP, Anda akan menemukan yang gratis dan terbuka:

- Alat dan standar keamanan aplikasi.
- Buku lengkap tentang pengujian keamanan aplikasi, pengembangan kode yang aman, dan tinjauan kode yang aman.
- Presentasi dan video.
- OWASP cheats tentang banyak topik umum.
- Kontrol keamanan dan pustaka standar.
- Chapter lokal di seluruh dunia.
- Penelitian mutakhir.
- Konferensi luas di seluruh dunia.
- Mailing list.

Pelajari lebih lanjut di: https://www.owasp.org.

Semua alat OWASP, dokumen, video, presentasi, dan bab bebas dan terbuka untuk siapa saja yang tertarik meningkatkan keamanan aplikasi.

Kami menganjurkan pendekatan keamanan aplikasi sebagai masalah orang, proses, dan teknologi, karena pendekatan keamanan aplikasi paling efektif memerlukan peningkatan di bidang ini.

OWASP adalah jenis organisasi baru. Kebebasan kami dari tekanan komersial memungkinkan kami untuk menyediakan informasi keamanan aplikasi yang tidak memihak, praktis, dan hemat biaya.

OWASP tidak berafiliasi dengan perusahaan teknologi mana pun, meskipun kami mendukung penggunaan teknologi keamanan komersial yang cerdas. OWASP memproduksi banyak jenis bahan dengan cara kolaboratif, transparan, dan terbuka.

Yayasan OWASP adalah entitas nirlaba yang memastikan kesuksesan jangka panjang proyek. Hampir semua orang yang terkait dengan OWASP adalah relawan, termasuk dewan OWASP, pemimpin chapter, pemimpin proyek, dan anggota proyek. Kami mendukung penelitian keamanan inovatif dengan hibah dan infrastruktur.

Ayo bergabung dengan kami!

Kata Pengantar

Elemen dasar inovasi di dunia aplikasi yang digerakkan saat ini adalah Antarmuka Pemrograman Aplikasi (API). Dari bank, ritel, dan transportasi hingga IoT, kendaraan otonom, dan kota pintar, API merupakan bagian penting dari aplikasi seluler, SaaS, dan web modern dan dapat ditemukan di aplikasi menghadap pelanggan, menghadap mitra, dan internal.

Karena sifatnya, API mengekspos logika aplikasi dan data sensitif seperti Informasi Pribadi (PII) dan karena itu, API semakin menjadi target para penyerang. Tanpa API yang aman, inovasi cepat akan mustahil.

Meskipun risiko keamanan web aplikasi yang lebih luas Top 10 masih masuk akal, karena sifat khusus mereka, daftar risiko keamanan API spesifik diperlukan. Keamanan API berfokus pada strategi dan solusi untuk memahami dan mengurangi kerentanan dan risiko keamanan yang unik terkait dengan API.

Jika Anda sudah familiar dengan Proyek OWASP Top 10, maka Anda akan melihat kesamaan antara kedua dokumen: keduanya ditujukan untuk keterbacaan dan adopsi. Jika Anda baru mengenal seri OWASP Top 10, mungkin lebih baik membaca bagian Risiko Keamanan API dan Metodologi dan Data sebelum melompat ke daftar 10 besar.

Anda dapat berkontribusi pada OWASP API Security Top 10 dengan pertanyaan, komentar, dan ide Anda di repositori proyek GitHub kami:

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

Anda dapat menemukan OWASP API Security Top 10 di sini:

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

Kami ingin berterima kasih kepada semua kontributor yang membuat proyek ini dimungkinkan dengan upaya dan kontribusi mereka. Mereka semua tercantum di bagian Ucapan Terima Kasih. Terima kasih!

Pendahuluan

Selamat Datang di OWASP API Security Top 10 - 2019!

Selamat datang di edisi pertama OWASP API Security Top 10. Jika Anda sudah akrab dengan seri OWASP Top 10, Anda akan melihat kesamaannya: tujuannya adalah mudah dibaca dan diadopsi. Jika tidak, pertimbangkan untuk mengunjungi halaman wiki OWASP API Security Project, sebelum menggali lebih dalam tentang risiko keamanan API yang paling kritis.

API memainkan peran yang sangat penting dalam arsitektur aplikasi modern. Karena penciptaan kesadaran keamanan dan inovasi memiliki tempo yang berbeda, penting untuk fokus pada kelemahan keamanan API yang umum.

Tujuan utama OWASP API Security Top 10 adalah untuk mendidik mereka yang terlibat dalam pengembangan dan pemeliharaan API, misalnya pengembang, desainer, arsitek, manajer, atau organisasi.

Di bagian Metodologi dan Data, Anda dapat membaca lebih lanjut tentang bagaimana edisi pertama ini dibuat. Di versi mendatang, kami ingin melibatkan industri keamanan, dengan panggilan data publik. Untuk saat ini, kami mendorong semua orang untuk berkontribusi dengan pertanyaan, komentar, dan ide di repositori GitHub atau Mailing list kami.

Catatan Rilis

Ini adalah edisi OWASP API Security Top 10 pertama, yang direncanakan akan diperbarui secara berkala, setiap tiga atau empat tahun sekali.

Tidak seperti versi ini, di versi mendatang, kami ingin melakukan panggilan data publik, melibatkan industri keamanan dalam upaya ini. Di bagian Metodologi dan Data, Anda akan menemukan detail lebih lanjut tentang bagaimana versi ini dibangun. Untuk detail lebih lanjut tentang risiko keamanan, harap merujuk pada bagian API Security Risks.

Penting untuk menyadari bahwa selama beberapa tahun terakhir, arsitektur aplikasi telah berubah secara signifikan. Saat ini, API memainkan peran yang sangat penting dalam arsitektur baru ini dari mikroservis, Single Page Applications (SPAs), aplikasi seluler, IoT, dan lainnya.

OWASP API Security Top 10 merupakan upaya yang dibutuhkan untuk menciptakan kesadaran tentang masalah keamanan API modern. Hanya mungkin karena upaya besar dari beberapa sukarelawan, semuanya terdaftar di bagian Penghargaan. Terima kasih!

Risiko Keamanan API

OWASP Risk Rating Methodology digunakan untuk melakukan analisis risiko.

Tabel di bawah ini merangkum terminologi yang terkait dengan skor risiko.

Agen Ancaman	Dapat Dieksploitasi	Prevalensi Kelemahan	Dapat Dideteksi Kelemahan	Dampak Teknis	Dampak Bisnis
Khusus API	Mudah: **3**	Luas **3**	Mudah **3**	Parah **3**	Spesifik Bisnis
Khusus API	Rata-rata: **2**	Umum **2**	Rata-rata **2**	Sedang **2**	Spesifik Bisnis
Khusus API	Sulit: **1**	Sulit **1**	Sulit **1**	Minor **1**	Spesifik Bisnis

Catatan: Pendekatan ini tidak memperhitungkan kemungkinan agen ancaman. Juga tidak memperhitungkan berbagai detail teknis yang terkait dengan aplikasi tertentu Anda. Faktor-faktor apa pun dapat secara signifikan mempengaruhi kemungkinan keseluruhan penyerang menemukan dan mengeksploitasi kerentanan tertentu. Peringkat ini tidak memperhitungkan dampak aktual pada bisnis Anda. Organisasi Anda harus memutuskan seberapa banyak risiko keamanan dari aplikasi dan API yang akan diterima organisasi mengingat budaya, industri, dan lingkungan peraturan Anda. Tujuan OWASP API Security Top 10 bukan untuk melakukan analisis risiko ini untuk Anda.

Referensi

OWASP

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

Eksternal

- 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 Top 10 Risiko Keamanan API – 2019

Risiko	Deskripsi
API1:2019 - Otorisasi Tingkat Objek yang Rusak	API cenderung mengekspos endpoint yang menangani pengidentifikasi objek, menciptakan masalah Otorisasi Akses Tingkat yang luas. Pemeriksaan otorisasi tingkat objek harus dipertimbangkan dalam setiap fungsi yang mengakses sumber data menggunakan input dari pengguna.
API2:2019 - Otentikasi Pengguna yang Rusak	Mekanisme otentikasi sering diimplementasikan dengan salah, memungkinkan penyerang mengkompromikan token otentikasi atau memanfaatkan celah implementasi untuk mengambil alih identitas pengguna lain sementara atau permanen. Mengompromikan kemampuan sistem untuk mengidentifikasi klien/pengguna, mengkompromikan keamanan API secara keseluruhan.
API3:2019 - Pemaparan Data yang Berlebihan	Menantikan implementasi generik, pengembang cenderung mengekspos semua properti objek tanpa mempertimbangkan sensitivitas individu mereka, mengandalkan klien untuk melakukan penyaringan data sebelum menampilkannya ke pengguna.
API4:2019 - Kurangnya Sumber Daya & Pembatasan Laju	Cukup sering, API tidak memberlakukan pembatasan apa pun pada ukuran atau jumlah sumber daya yang dapat diminta oleh klien/pengguna. Tidak hanya dapat berdampak pada kinerja server API, yang mengarah ke Denial of Service (DoS), tetapi juga membiarkan pintu terbuka untuk celah otentikasi seperti brute force.
API5:2019 - Otorisasi Tingkat Fungsi yang Rusak	Kebijakan kontrol akses kompleks dengan hierarki, kelompok, dan peran yang berbeda, dan pemisahan yang tidak jelas antara fungsi administratif dan reguler, cenderung mengarah pada celah otorisasi. Dengan memanfaatkan masalah ini, penyerang mendapatkan akses ke sumber daya pengguna lain dan/atau fungsi administratif.
API6:2019 - Mass Assignment	Mengikat data yang disediakan klien (misalnya, JSON) ke model data, tanpa penyaringan properti yang tepat berdasarkan daftar putih, biasanya mengarah ke Mass Assignment. Menebak properti objek, mengeksplorasi titik akhir API lainnya, membaca dokumentasi, atau menyediakan properti objek tambahan dalam muatan permintaan, memungkinkan penyerang memodifikasi properti objek yang seharusnya tidak mereka lakukan.

API7:2019 - Kesalahan Konfigurasi Keamanan	Kesalahan konfigurasi keamanan umumnya merupakan hasil dari konfigurasi default yang tidak aman, konfigurasi yang tidak lengkap atau ad-hoc, penyimpanan cloud terbuka, header HTTP yang dikonfigurasi salah, metode HTTP yang tidak perlu, berbagi sumber daya Cross-Origin (CORS) yang longgar, dan pesan kesalahan yang terperinci mengandung informasi sensitif.
API8:2019 - Injeksi	Celah injeksi, seperti SQL, NoSQL, Command Injection, dll., terjadi ketika data yang tidak dipercaya dikirim ke interpreter sebagai bagian dari perintah atau kueri. Data berbahaya penyerang dapat menipu interpreter untuk mengeksekusi perintah yang tidak diinginkan atau mengakses data tanpa otorisasi yang tepat.
API9:2019 - Pengelolaan Aset yang Tidak Tepat	API cenderung mengekspos lebih banyak endpoint daripada aplikasi web tradisional, sehingga dokumentasi yang tepat dan terbaru sangat penting. Inventarisasi host dan versi API yang diterapkan yang tepat juga memainkan peran penting untuk memitigasi masalah seperti versi API usang dan endpoint debug yang terekspos.
API10:2019 - Logging & Pemantauan yang Tidak Memadai	Pencatatan dan pemantauan yang tidak memadai, dipasangkan dengan integrasi insiden yang hilang atau tidak efektif, memungkinkan penyerang untuk menyerang sistem lebih lanjut, mempertahankan persistensi, berpindah ke lebih banyak sistem untuk mengutak-atik, mengekstrak, atau menghancurkan data. Sebagian besar studi pelanggaran menunjukkan waktu untuk mendeteksi pelanggaran adalah lebih dari 200 hari, biasanya dideteksi oleh pihak eksternal daripada proses internal atau pemantauan.

API1:2019 Otorisasi Tingkat Objek yang Rusak

Agen Ancaman/Vektor Serangan	Kelemahan Keamanan	Dampak
Khusus API: Eksploitasi 3	Prevalensi **3** : Deteksi **2**	Teknis **3** : Spesifik Bisnis
Penyerang dapat memanfaatkan endpoint API yang rentan terhadap otorisasi tingkat objek yang rusak dengan memanipulasi ID objek yang dikirim dalam permintaan. Hal ini dapat menyebabkan akses tidak sah ke data sensitif. Masalah ini sangat umum dalam aplikasi berbasis API karena komponen server biasanya tidak sepenuhnya melacak status klien, dan sebaliknya, lebih bergantung pada parameter seperti ID objek, yang dikirim dari klien untuk memutuskan objek mana yang akan diakses.	Ini telah menjadi serangan paling umum dan berdampak pada API. Mekanisme otorisasi dan kontrol akses dalam aplikasi modern kompleks dan meluas. Bahkan jika aplikasi mengimplementasikan infrastruktur yang tepat untuk pemeriksaan otorisasi, pengembang mungkin lupa menggunakan pemeriksaan ini sebelum mengakses objek sensitif. Deteksi kontrol akses biasanya tidak dapat diterapkan untuk pengujian statis atau dinamis otomatis.	Akses tidak sah dapat mengakibatkan pengungkapan data ke pihak yang tidak berwenang, kehilangan data, atau manipulasi data. Akses tidak sah ke objek juga dapat mengarah ke pengambilalihan akun secara penuh.

Apakah API Rentan?

Otorisasi tingkat objek adalah mekanisme kontrol akses yang biasanya diimplementasikan di tingkat kode untuk memvalidasi bahwa satu pengguna hanya dapat mengakses objek yang seharusnya mereka akses.

Setiap endpoint API yang menerima ID objek, dan melakukan jenis tindakan apa pun pada objek, harus menerapkan pemeriksaan otorisasi tingkat objek. Pemeriksaan harus memvalidasi bahwa pengguna yang login memiliki akses untuk melakukan tindakan yang diminta pada objek yang diminta.

Kegagalan dalam mekanisme ini biasanya menyebabkan pengungkapan informasi yang tidak sah, modifikasi, atau penghancuran semua data.

Skenario Serangan Contoh

Skenario #1

Platform e-commerce untuk toko online (toko) menyediakan halaman daftar dengan grafik pendapatan untuk toko hosting mereka. Memeriksa permintaan browser, penyerang dapat mengidentifikasi endpoint API yang digunakan sebagai sumber data untuk grafik tersebut dan polanya /shops/{shopName}/revenue_data.json. Menggunakan endpoint API lainnya, penyerang dapat mendapatkan daftar semua nama toko yang di-host. Dengan skrip sederhana untuk memanipulasi nama di daftar, mengganti {shopName} dalam URL, penyerang mendapatkan akses ke data penjualan ribuan toko e-commerce.

Skenario #2

Saat memantau lalu lintas jaringan perangkat wearable, permintaan HTTP PATCH berikut menarik perhatian penyerang karena adanya header permintaan HTTP kustom X-User-Id: 54796. Mengganti nilai X-User-Id dengan 54795, penyerang menerima respons HTTP yang berhasil, dan dapat memodifikasi data akun pengguna lain.

Cara Mencegah

- Implementasikan mekanisme otorisasi yang tepat yang mengandalkan kebijakan dan hierarki pengguna.
- Gunakan mekanisme otorisasi untuk memeriksa apakah pengguna yang login memiliki akses untuk melakukan tindakan yang diminta pada catatan di setiap fungsi yang menggunakan input dari klien untuk mengakses catatan di database.
- Lebih baik menggunakan nilai acak dan tidak terduga sebagai ID catatan.
- Menulis tes untuk mengevaluasi mekanisme otorisasi. Jangan menerapkan perubahan rentan yang merusak tes.

Referensi

Eksternal

- CWE-284: Kontrol Akses yang Tidak Tepat
- CWE-285: Otorisasi yang Tidak Tepat
- CWE-639: Otorisasi Melewati Kunci yang Dikendalikan Pengguna

API2:2019 Otentikasi Pengguna yang Rusak

Agen Ancaman/Vektor Serangan	Kelemahan Keamanan	Dampak
Khusus API: Eksploitasi 3	Prevalensi **2** : Deteksi **2**	Teknis **3** : Spesifik Bisnis
Otentikasi dalam API adalah mekanisme yang kompleks dan membingungkan. Insinyur perangkat lunak dan keamanan mungkin memiliki kesalahpahaman tentang batasan otentikasi dan cara mengimplementasikannya dengan benar. Selain itu, mekanisme otentikasi adalah target yang mudah bagi penyerang, karena terbuka untuk semua orang. Dua poin ini membuat komponen otentikasi berpotensi rentan terhadap banyak eksploitasi.	Ada dua sub-masalah: 1. Kurangnya mekanisme perlindungan: endpoint API yang bertanggung jawab untuk otentikasi harus diperlakukan berbeda dari endpoint reguler dan menerapkan lapisan perlindungan tambahan 2. Kesalahan implementasi mekanisme: Mekanisme digunakan/diimplementasikan tanpa mempertimbangkan vektor serangan, atau itu kasus penggunaan yang salah (misalnya, mekanisme otentikasi yang dirancang untuk klien IoT mungkin bukan pilihan yang tepat untuk aplikasi web).	Penyerang dapat mengambil alih akun pengguna lain dalam sistem, membaca data pribadi mereka, dan melakukan tindakan sensitif atas nama mereka, seperti transaksi uang dan mengirim pesan pribadi.

Apakah API Rentan?

Titik akhir dan alur otentikasi adalah aset yang perlu dilindungi. “Lupa kata sandi / reset kata sandi” harus diperlakukan sama seperti mekanisme otentikasi.

API rentan jika:

- Mengizinkan credential stuffing di mana penyerang memiliki daftar nama pengguna dan kata sandi yang valid.
- Mengizinkan penyerang melakukan serangan brute force pada akun pengguna yang sama, tanpa menyajikan mekanisme captcha/penguncian akun.
- Mengizinkan kata sandi yang lemah.
- Mengirim detail otentikasi sensitif, seperti token otentikasi dan kata sandi di URL.
- Tidak memvalidasi keaslian token.
- Menerima token JWT yang tidak ditandatangani/lemah ditandatangani (`"alg": "none"`) / tidak memvalidasi tanggal kedaluwarsa mereka.

- Menggunakan kata sandi teks polos, tidak dienkripsi, atau di-hash lemah.
- Menggunakan kunci enkripsi yang lemah.

Skenario Serangan Contoh

Skenario #1

Credential stuffing (menggunakan daftar nama pengguna/kata sandi yang diketahui), adalah serangan yang umum. Jika aplikasi tidak menerapkan ancaman otomatis atau perlindungan stuffing kredensial, aplikasi dapat digunakan sebagai oracle kata sandi (penguji) untuk menentukan apakah kredensial valid.

Skenario #2

Seorang penyerang memulai alur kerja pemulihan kata sandi dengan menerbitkan permintaan POST ke /api/system/verification-codes dan dengan menyediakan nama pengguna dalam body permintaan. Selanjutnya token SMS dengan 6 digit dikirim ke telepon korban. Karena API tidak menerapkan kebijakan pembatasan laju, penyerang dapat menguji semua kombinasi yang mungkin menggunakan skrip multi-thread, terhadap endpoint /api/system/verification-codes/{smsToken} untuk menemukan token yang benar dalam beberapa menit.

Cara Mencegah

- Pastikan Anda mengetahui semua kemungkinan alur untuk mengotentikasi ke API (mobile/web/tautan dalam yang mengimplementasikan otentikasi satu klik/dll.)
- Tanyakan pada insinyur Anda alur apa yang Anda lewatkan.
- Baca tentang mekanisme otentikasi Anda. Pastikan Anda memahami apa dan bagaimana mereka digunakan. OAuth bukan otentikasi, dan begitu juga kunci API.
- Jangan menemukan kembali roda dalam otentikasi, generasi token, penyimpanan kata sandi. Gunakan standar.
- Endpoint pemulihan kredensial/lupa kata sandi harus diperlakukan seperti titik akhir login dalam hal brute force, pembatasan laju, dan perlindungan penguncian.
- Gunakan OWASP Authentication Cheatsheet.
- Jika memungkinkan, terapkan otentikasi multifaktor.
- Terapkan mekanisme anti-brute force untuk memitigasi stuffing kredensial, serangan kamus, dan serangan brute force pada titik akhir otentikasi Anda. Mekanisme ini harus lebih ketat daripada mekanisme pembatasan laju normal pada API Anda.
- Terapkan penguncian akun / mekanisme captcha untuk mencegah brute force terhadap pengguna tertentu. Terapkan pemeriksaan kata sandi lemah.

- Kunci API seharusnya tidak digunakan untuk otentikasi pengguna, tetapi untuk otentikasi aplikasi/proyek klien.

Referensi

OWASP

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

Eksternal

- CWE-798: Penggunaan Kredensial Hard-coded

API3:2019 Pemaparan Data yang Berlebihan

Agen Ancaman/Vektor Serangan	Kelemahan Keamanan	Dampak
Khusus API: Eksploitasi 3	Prevalensi **2** : Deteksi **2**	Teknis **2** : Spesifik Bisnis
Eksploitasi Pemaparan Data Berlebihan sederhana, dan biasanya dilakukan dengan menyadap lalu lintas untuk menganalisis respon API, mencari pemaparan data sensitif yang seharusnya tidak dikembalikan ke pengguna.	API mengandalkan klien untuk melakukan penyaringan data. Karena API digunakan sebagai sumber data, terkadang pengembang mencoba mengimplementasikannya secara generik tanpa memikirkan sensitivitas data yang terpapar. Alat otomatis biasanya tidak dapat mendeteksi jenis kerentanan ini karena sulit membedakan antara data yang sah dikembalikan dari API, dan data sensitif yang tidak boleh dikembalikan tanpa pemahaman mendalam tentang aplikasi.	Pemaparan Data Berlebihan umumnya mengarah pada pemaparan data sensitif.

Apakah API Rentan?

API mengembalikan data sensitif ke klien berdasarkan desain. Data ini biasanya disaring di sisi klien sebelum ditampilkan ke pengguna. Penyerang dengan mudah dapat menyadap lalu lintas dan melihat data sensitif.

Skenario Serangan Contoh

Skenario #1

Tim seluler menggunakan endpoint `/api/articles/{articleId}/comments/{commentId}` dalam tampilan artikel untuk merender metadata komentar. Menyadap lalu lintas aplikasi seluler, seorang penyerang mengetahui bahwa data sensitif lain terkait penulis komentar juga dikembalikan. Implementasi endpoint menggunakan metode `toJSON()` generik pada model `User`, yang berisi PII, untuk men-serialisasi objek.

Skenario #2

Sistem pengawasan berbasis IOT memungkinkan administrator membuat pengguna dengan izin yang berbeda. Seorang admin membuat akun pengguna untuk satpam baru yang hanya boleh mengakses bangunan tertentu di situs tersebut. Setelah satpam menggunakan aplikasi selulernya, panggilan API dipicu ke: `/api/sites/111/cameras` untuk menerima data tentang kamera yang tersedia dan menampilkannya di dashboard. Respons berisi daftar dengan rincian tentang kamera dalam format berikut: `{"id":"xxx","live_access_token":"xxxx-bbbbb","building_id":"yyy"}`. Meskipun GUI klien hanya menampilkan kamera yang seharusnya satpam ini akses, respons API aktual berisi daftar lengkap semua kamera di situs.

Cara Mencegah

- Jangan pernah mengandalkan sisi klien untuk menyaring data sensitif.
- Tinjau respon dari API untuk memastikan hanya berisi data yang sah.
- Insinyur backend harus selalu bertanya pada diri sendiri "siapa konsumen data ini?" sebelum memaparkan endpoint API baru.
- Hindari menggunakan metode generik seperti `to_json()` dan `to_string()`. Sebaliknya, pilih properti spesifik yang benar-benar ingin Anda kembalikan.
- Klasifikasikan informasi sensitif dan pribadi (PII) yang disimpan dan dikelola aplikasi Anda, meninjau semua panggilan API yang mengembalikan informasi tersebut untuk melihat apakah respons ini menimbulkan masalah keamanan.
- Implementasikan mekanisme validasi respons berbasis skema sebagai lapisan keamanan tambahan. Sebagai bagian dari mekanisme ini, tentukan dan paksakan data yang dikembalikan oleh semua metode API, termasuk kesalahan.

Referensi

Eksternal

- CWE-213: Pemaparan Informasi yang Disengaja

API4:2019 Kurangnya Sumber Daya & Pembatasan Laju

Agen Ancaman/Vektor Serangan	Kelemahan Keamanan	Dampak
Khusus API: Eksploitasi 2	Prevalensi **3** : Deteksi **3**	Teknis **2** : Spesifik Bisnis
Eksploitasi memerlukan permintaan API sederhana. Tidak ada otentikasi yang diperlukan. Beberapa permintaan secara bersamaan dapat dilakukan dari satu komputer lokal atau dengan menggunakan sumber daya komputasi cloud.	Umum ditemukan API yang tidak menerapkan pembatasan laju atau API di mana batas tidak ditetapkan dengan benar.	Eksploitasi dapat mengarah ke DoS, membuat API tidak responsif atau bahkan tidak tersedia.

Apakah API Rentan?

Permintaan API mengonsumsi sumber daya seperti jaringan, CPU, memori, dan penyimpanan. Jumlah sumber daya yang diperlukan untuk memenuhi permintaan sangat bergantung pada input pengguna dan logika bisnis endpoint. Juga, pertimbangkan fakta bahwa permintaan dari beberapa klien API bersaing untuk sumber daya. API rentan jika setidaknya satu dari batasan berikut hilang atau disetel secara tidak tepat (misalnya, terlalu rendah/tinggi):

- Batas waktu eksekusi
- Memori maksimum yang dapat dialokasikan
- Jumlah deskriptor berkas
- Jumlah proses
- Ukuran muatan permintaan (misalnya, unggahan)
- Jumlah permintaan per klien/sumber daya
- Jumlah catatan per halaman untuk dikembalikan dalam satu respons permintaan

Skenario Serangan Contoh

Skenario #1

Seorang penyerang mengunggah gambar besar dengan menerbitkan permintaan POST ke `/api/v1/images`. Saat unggahan selesai, API membuat beberapa thumbnail dengan ukuran yang berbeda. Karena ukuran gambar yang diunggah, memori yang tersedia habis selama pembuatan thumbnail dan API menjadi tidak responsif.

Skenario #2

Kami memiliki aplikasi yang berisi daftar pengguna di UI dengan batas `200` pengguna per halaman. Daftar pengguna diambil dari server menggunakan kueri berikut: `/api/users?page=1&size=200`. Seorang penyerang mengubah parameter `size` menjadi `200.000`, menyebabkan masalah kinerja pada basis data. Sementara itu, API menjadi tidak responsif dan tidak dapat menangani permintaan lebih lanjut dari klien ini atau klien lainnya (alias DoS).
Skenario yang sama dapat digunakan untuk memancing kesalahan Integer Overflow atau Buffer Overflow.

Cara Mencegah

- Docker memudahkan untuk membatasi memori, CPU, jumlah restart, deskriptor berkas, dan proses.
- Terapkan batas seberapa sering klien dapat memanggil API dalam rentang waktu tertentu.
- Beri tahu klien saat batas terlampaui dengan menyediakan nomor batas dan waktu saat batas akan direset.
- Tambahkan validasi server-side yang tepat untuk parameter string kueri dan body permintaan, khususnya yang mengendalikan jumlah catatan yang akan dikembalikan dalam respons.
- Tentukan dan tegakkan ukuran maksimum data pada semua parameter dan muatan masukan seperti panjang maksimum untuk string dan jumlah elemen maksimum dalam array.

Referensi

- Blocking Brute Force Attacks
- Docker Cheat Sheet - Limit resources (memory, CPU, file descriptors, processes, restarts)
- REST Assessment Cheat Sheet

Eksternal

- CWE-307: Improper Restriction of Excessive Authentication Attempts
- CWE-770: Allocation of Resources Without Limits or Throttling
- "*Rate Limiting (Throttling)*" - Security Strategies for Microservices-based Application Systems, NIST

API5:2019 Otorisasi Tingkat Fungsi yang Rusak

Agen Ancaman/Vektor Serangan	Kelemahan Keamanan	Dampak
Khusus API: Eksploitasi 3	Prevalensi **2** : Deteksi **1**	Teknis **2** : Spesifik Bisnis
Eksploitasi membutuhkan penyerang untuk mengirim panggilan API yang sah ke endpoint API yang seharusnya tidak mereka akses. Endpoint ini mungkin terbuka untuk pengguna anonim atau pengguna reguler non-istimewa. Lebih mudah menemukan celah ini di API karena API lebih terstruktur, dan cara mengakses fungsi tertentu lebih dapat diprediksi (misalnya, mengganti metode HTTP dari GET ke PUT, atau mengubah string "users" di URL menjadi "admins").	Pemeriksaan otorisasi untuk fungsi atau sumber daya biasanya dikelola melalui konfigurasi, dan terkadang di tingkat kode. Mengimplementasikan pemeriksaan yang tepat dapat menjadi tugas yang membingungkan, karena aplikasi modern dapat berisi banyak jenis peran atau kelompok dan hirarki pengguna yang kompleks (misalnya, sub-pengguna, pengguna dengan lebih dari satu peran).	Celah seperti itu memungkinkan penyerang mengakses fungsionalitas yang tidak sah. Fungsi administratif menjadi target utama untuk jenis serangan ini.

Apakah API Rentan?

Cara terbaik untuk menemukan masalah otorisasi tingkat fungsi yang rusak adalah dengan melakukan analisis mendalam terhadap mekanisme otorisasi, dengan mempertimbangkan hirarki pengguna, peran atau kelompok yang berbeda dalam aplikasi, dan mengajukan pertanyaan berikut:

- Apakah pengguna reguler dapat mengakses endpoint administratif?
- Apakah pengguna dapat melakukan tindakan sensitif (misalnya, pembuatan, modifikasi, atau penghapusan) yang seharusnya tidak mereka akses dengan hanya mengubah metode HTTP (misalnya, dari GET ke DELETE)?
- Apakah pengguna dari kelompok X dapat mengakses fungsi yang seharusnya hanya diekspos ke pengguna dari kelompok Y, dengan hanya menebak URL dan parameter endpoint (misalnya, /api/v1/users/export_all)?

Jangan menganggap endpoint API adalah reguler atau administratif hanya berdasarkan jalur URL.

Meskipun pengembang mungkin memilih untuk mengekspos sebagian besar endpoint administratif di bawah jalur relatif tertentu, seperti api/admins, sangat umum menemukan endpoint administratif ini di bawah jalur relatif lain bersama dengan endpoint reguler, seperti api/users.

Skenario Serangan Contoh

Skenario #1

Selama proses pendaftaran ke aplikasi yang hanya mengizinkan pengguna diundang untuk bergabung, aplikasi seluler memicu panggilan API ke GET /api/invites/{invite_guid}. Respons berisi JSON dengan rincian undangan, termasuk peran pengguna dan email pengguna.
Seorang penyerang menduplikasi permintaan dan memanipulasi metode HTTP dan endpoint menjadi POST /api/invites/new. Endpoint ini hanya boleh diakses oleh administrator menggunakan konsol admin, yang tidak menerapkan pemeriksaan otorisasi tingkat fungsi.
Penyerang mengeksploitasi masalah ini dan mengirim undangan ke dirinya sendiri untuk membuat akun admin:

```
POST /api/invites/new

{"email":"hugo@malicious.com","role":"admin"}
```

Skenario #2

Sebuah API berisi endpoint yang seharusnya hanya diekspos ke administrator - `GET /api/admin/v1/users/all`. Endpoint ini mengembalikan rincian semua pengguna aplikasi dan tidak menerapkan pemeriksaan otorisasi tingkat fungsi. Seorang penyerang yang mempelajari struktur API membuat tebakan terdidik dan berhasil mengakses endpoint ini, yang mengekspos rincian sensitif pengguna aplikasi.

Cara Mencegah

Aplikasi Anda harus memiliki modul otorisasi yang konsisten dan mudah dianalisis yang dipanggil dari semua fungsi bisnis Anda. Seringkali, perlindungan tersebut disediakan oleh satu atau lebih komponen eksternal untuk kode aplikasi.

- Mekanisme penegakan harus menolak semua akses secara default, membutuhkan izin eksplisit ke peran tertentu untuk mengakses setiap fungsi.
- Tinjau endpoint API Anda terhadap celah otorisasi tingkat fungsi, dengan mempertimbangkan logika bisnis aplikasi dan hirarki kelompok.
- Pastikan semua pengendali administrasi Anda mewarisi pengendali abstrak administratif yang menerapkan pemeriksaan otorisasi berdasarkan grup/peran pengguna.
- Pastikan fungsi administratif di dalam pengendali reguler menerapkan pemeriksaan otorisasi berdasarkan grup dan peran pengguna.

Referensi

OWASP

- Artikel OWASP tentang Forced Browsing
- OWASP Top 10 2013-A7-Missing Function Level Access Control
- OWASP Development Guide: Bab tentang Otorisasi

Eksternal

- CWE-285: Otorisasi yang Tidak Tepat

API6:2019 - Mass Assignment

Agen Ancaman/Vektor Serangan	Kelemahan Keamanan	Dampak
Khusus API: Eksploitasi 2	Prevalensi **2** : Deteksi **2**	Teknis **2** : Spesifik Bisnis
Eksploitasi biasanya memerlukan pemahaman tentang logika bisnis, hubungan objek, dan struktur API. Eksploitasi penugasan massal lebih mudah dalam API, karena secara desain mereka mengekspos implementasi aplikasi yang mendasari beserta nama properti.	Kerangka kerja modern mendorong pengembang untuk menggunakan fungsi yang secara otomatis mengikat masukan dari klien ke dalam variabel kode dan objek internal. Penyerang dapat menggunakan metodologi ini untuk memperbarui atau menimpa properti objek sensitif yang sebenarnya tidak dimaksudkan untuk diekspos oleh pengembang.	Eksploitasi dapat menyebabkan eskalasi hak istimewa, perusakan data, menghindari mekanisme keamanan, dan lainnya.

Apakah API Rentan?

Objek dalam aplikasi modern mungkin berisi banyak properti. Beberapa properti ini harus diperbarui langsung oleh klien (misalnya, `user.first_name` atau `user.address`) dan beberapa tidak boleh (misalnya, flag `user.is_vip`).
Titik akhir API rentan jika secara otomatis mengubah parameter klien menjadi properti objek internal, tanpa mempertimbangkan sensitivitas dan tingkat paparan properti tersebut. Hal ini bisa memungkinkan penyerang untuk memperbarui properti objek yang seharusnya tidak mereka akses.

Contoh properti sensitif:

- **Properti terkait izin**: `user.is_admin`, `user.is_vip` hanya boleh diatur oleh admin.
- **Properti tergantung proses**: `user.cash` hanya boleh diatur secara internal setelah verifikasi pembayaran.
- **Properti internal**: `article.created_time` hanya boleh diatur secara internal oleh aplikasi.

Skenario Serangan Contoh

Skenario #1

Aplikasi berbagi tumpangan memberi pengguna opsi untuk mengedit informasi dasar untuk profil mereka. Selama proses ini, panggilan API dikirim ke `PUT /api/v1/users/me` dengan objek JSON yang sah:

```
{"user_name":"inons","age":24}
```

Permintaan GET /api/v1/users/me menyertakan properti credit_balance tambahan:
{"user_name":"inons","age":24,"credit_balance":10}
Penyerang memutar ulang permintaan pertama dengan payload berikut:

{"user_name":"attacker","age":60,"credit_balance":99999}
Karena endpoint rentan terhadap penugasan massal, penyerang menerima kredit tanpa membayar.

Skenario #2

Portal berbagi video memungkinkan pengguna mengunggah konten dan mengunduh konten dalam format yang berbeda. Seorang penyerang yang menjelajahi API menemukan bahwa endpoint GET /api/v1/videos/{video_id}/meta_data mengembalikan objek JSON dengan properti video. Salah satu propertinya adalah "mp4_conversion_params":"-v codec h264" yang menunjukkan bahwa aplikasi menggunakan perintah shell untuk mengubah video.
Penyerang juga menemukan endpoint POST /api/v1/videos/new rentan terhadap penugasan massal dan memungkinkan klien mengatur properti apa pun dari objek video. Penyerang menetapkan nilai berbahaya sebagai berikut: "mp4_conversion_params":"-v codec h264 && format C:/". Nilai ini akan menyebabkan injeksi perintah shell setelah penyerang mengunduh video sebagai MP4.

Cara Mencegah

- Jika memungkinkan, hindari menggunakan fungsi yang secara otomatis mengikat masukan klien ke dalam variabel kode atau objek internal.
- Daftar putih hanya properti yang seharusnya diperbarui oleh klien.
- Gunakan fitur bawaan untuk daftar hitam properti yang tidak boleh diakses oleh klien.
- Jika berlaku, tentukan dan tegakkan secara eksplisit skema untuk muatan data masukan.

Referensi

Eksternal

- CWE-915: Pengontrolan yang Tidak Tepat dari Modifikasi Atribut Objek yang Ditentukan Secara Dinamis

API7:2019 Kesalahan Konfigurasi Keamanan

Agen Ancaman/Vektor Serangan	Kelemahan Keamanan	Dampak
Khusus API: Eksploitasi 3	Prevalensi **3** : Deteksi **3**	Teknis **2** : Spesifik Bisnis
Penyerang sering mencoba menemukan celah yang tidak diperbarui, endpoint umum, atau file dan direktori yang tidak dilindungi untuk mendapatkan akses yang tidak sah atau pengetahuan tentang sistem.	Kesalahan konfigurasi keamanan dapat terjadi pada setiap level tumpukan API, dari level jaringan hingga level aplikasi. Alat otomatis tersedia untuk mendeteksi dan memanfaatkan kesalahan konfigurasi seperti layanan yang tidak perlu atau opsi warisan.	Kesalahan konfigurasi keamanan tidak hanya dapat mengekspos data pengguna yang sensitif, tetapi juga rincian sistem yang dapat mengarah ke kompromi server penuh.

Apakah API Rentan?

API mungkin rentan jika:

- Pengerasan keamanan yang tepat hilang di bagian mana pun dari tumpukan aplikasi, atau jika memiliki izin yang dikonfigurasi dengan salah pada layanan cloud.
- Perbaikan keamanan terbaru hilang, atau sistemnya sudah ketinggalan zaman.
- Fitur yang tidak perlu diaktifkan (misalnya, kata kerja HTTP).
- Keamanan Lapisan Transport (TLS) hilang.
- Direktif keamanan tidak dikirim ke klien (misalnya, Security Headers).
- Kebijakan Berbagi Sumber Daya Lintas Asal (CORS) hilang atau disetel dengan salah.
- Pesan kesalahan termasuk jejak tumpukan, atau informasi sensitif lainnya terekspos.

Skenario Serangan Contoh

Skenario #1

Seorang penyerang menemukan file .bash_history di bawah direktori root server, yang berisi perintah yang digunakan oleh tim DevOps untuk mengakses API:

```
$ curl -X GET 'https://api.server/endpoint/' -H 'authorization: Basic Zm9vOmJhcg=='
```

Penyerang juga bisa menemukan endpoint baru pada API yang hanya digunakan oleh tim DevOps dan tidak didokumentasikan.

Skenario #2

Untuk menargetkan layanan tertentu, seorang penyerang menggunakan mesin pencari populer untuk mencari komputer yang dapat diakses langsung dari Internet. Penyerang menemukan host yang menjalankan sistem manajemen basis data populer, mendengarkan di port default. Host tersebut menggunakan konfigurasi default, yang secara default menonaktifkan otentikasi, dan penyerang mendapatkan akses ke jutaan catatan dengan PII, preferensi pribadi, dan data otentikasi.

Skenario #3

Memeriksa lalu lintas aplikasi seluler, penyerang mengetahui bahwa tidak semua lalu lintas HTTP dilakukan pada protokol aman (misalnya, TLS). Penyerang menemukan ini benar, khususnya untuk mengunduh gambar profil. Karena interaksi pengguna bersifat biner, meskipun lalu lintas API dilakukan pada protokol yang aman, penyerang menemukan pola pada ukuran respons API, yang dia gunakan untuk melacak preferensi pengguna atas konten yang dirender (misalnya, gambar profil).

Cara Mencegah

Siklus hidup API harus mencakup:

- Proses pengerasan yang dapat diulang yang mengarah ke penyebaran yang cepat dan mudah dari lingkungan yang dikunci dengan benar.
- Tugas untuk meninjau dan memperbarui konfigurasi di seluruh tumpukan API. Tinjauan harus mencakup: file orchestrasi, komponen API, dan layanan cloud (misalnya, izin bucket S3).
- Saluran komunikasi yang aman untuk semua interaksi akses API ke aset statis (misalnya, gambar).
- Proses otomatis untuk secara kontinu menilai efektivitas konfigurasi dan pengaturan di semua lingkungan.

Selanjutnya:

- Untuk mencegah jejak pengecualian dan informasi berharga lainnya dikirim kembali ke penyerang, jika berlaku, tentukan dan tegakkan semua skema muatan respons API termasuk respons kesalahan.
- Pastikan API hanya dapat diakses oleh kata kerja HTTP yang ditentukan. Semua kata kerja HTTP lainnya harus dinonaktifkan (misalnya, HEAD).
- API yang diharapkan dapat diakses dari klien berbasis browser (misalnya, front-end WebApp) harus menerapkan kebijakan Berbagi Sumber Daya Lintas Asal (CORS) yang tepat.

Referensi

OWASP

- OWASP Secure Headers Project
- OWASP Testing Guide: Configuration Management
- OWASP Testing Guide: Testing for Error Codes
- OWASP Testing Guide: Test Cross Origin Resource Sharing

Eksternal

- CWE-2: Kelemahan Keamanan Lingkungan
- CWE-16: Konfigurasi
- CWE-388: Penanganan Kesalahan
- Panduan Keamanan Server Umum, NIST
- Let's Encrypt: Otoritas Sertifikat Gratis, Otomatis, dan Terbuka

API8:2019 Injeksi

Agen Ancaman/Vektor Serangan	Kelemahan Keamanan	Dampak
Khusus API: Eksploitasi 3	Prevalensi **2** : Deteksi **3**	Teknis **3** : Spesifik Bisnis
Penyerang akan memberi makan API dengan data berbahaya melalui vektor injeksi apa pun yang tersedia (misalnya, input langsung, parameter, layanan terintegrasi, dll.), berharap itu dikirim ke interpreter.	Celah injeksi sangat umum dan sering ditemukan dalam kueri SQL, LDAP, atau NoSQL, perintah OS, parser XML, dan ORM. Celah ini mudah ditemukan saat meninjau kode sumber. Penyerang dapat menggunakan scanner dan fuzzer.	Injeksi dapat menyebabkan pengungkapan informasi dan kehilangan data. Itu juga dapat menyebabkan DoS, atau pengambilalihan host secara total.

Apakah API Rentan?

API rentan terhadap celah injeksi jika:

- Data yang disediakan klien tidak divalidasi, difilter, atau disucihamakan oleh API.
- Data yang disediakan klien digunakan langsung atau digabungkan ke kueri SQL/NoSQL/LDAP, perintah OS, parser XML, dan Pemetaan Objek Relasional (ORM)/Pemetaan Dokumen Objek (ODM).
- Data yang berasal dari sistem eksternal (misalnya, sistem terintegrasi) tidak divalidasi, difilter, atau disucihamakan oleh API.

Skenario Serangan Contoh

Skenario #1

Firmware dari perangkat kontrol orang tua menyediakan endpoint /api/CONFIG/restore yang mengharapkan appId dikirim sebagai parameter multipart. Menggunakan dekompiler, seorang penyerang mengetahui bahwa appId dilewatkan langsung ke panggilan sistem tanpa pembersihan apa pun:

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

Perintah berikut memungkinkan penyerang mematikan perangkat mana pun dengan firmware yang sama yang rentan:

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

Skenario #2

Kami memiliki aplikasi dengan fungsionalitas CRUD dasar untuk operasi dengan pemesanan. Seorang penyerang berhasil mengidentifikasi bahwa injeksi NoSQL mungkin dimungkinkan melalui parameter string kueri `bookingId` dalam permintaan penghapusan pemesanan. Beginilah permintaannya: `DELETE /api/bookings?bookingId=678`.
Server API menggunakan fungsi berikut untuk menangani permintaan penghapusan:

```
router.delete('/bookings', async function (req, res, next) {
  try {
     const deletedBooking = await Bookings.findOneAndRemove({'_id' : req.query.bookingId});
     res.status(200);
  } catch (err) {
    res.status(400).json({error: 'Unexpected error occured while processing a request'});
  }
});
```

Penyerang menyadap permintaan dan mengubah parameter string kueri `bookingId` seperti di bawah ini. Dalam hal ini, penyerang berhasil menghapus pemesanan pengguna lain:
`DELETE /api/bookings?bookingId[$ne]=678`

Cara Mencegah

Mencegah injeksi memerlukan pemisahan data dari perintah dan kueri.

- Lakukan validasi data menggunakan satu pustaka yang tepercaya dan dikelola secara aktif.
- Validasi, filter, dan sucikan semua data yang disediakan klien, atau data lainnya yang berasal dari sistem terintegrasi.
- Karakter khusus harus dilepas menggunakan sintaks spesifik untuk interpreter tujuan.
- Lebih baik menggunakan API yang aman yang menyediakan antarmuka terparameter.
- Selalu batasi jumlah catatan yang dikembalikan untuk mencegah pengungkapan massal jika terjadi injeksi.
- Validasi data masuk menggunakan filter yang cukup untuk hanya mengizinkan nilai yang valid untuk setiap parameter input.
- Tentukan jenis data dan pola ketat untuk semua parameter string.

Referensi

OWASP

- OWASP Injection Flaws
- SQL Injection
- NoSQL Injection Fun with Objects and Arrays

- Command Injection

Eksternal

- CWE-77: Command Injection
- CWE-89: SQL Injection

API9:2019 Pengelolaan Aset yang Tidak Tepat

Agen Ancaman/Vektor Serangan	Kelemahan Keamanan	Dampak
Khusus API: Eksploitasi 3	Prevalensi **3** : Deteksi **2**	Teknis **2** : Spesifik Bisnis
Versi API lama biasanya tidak diperbarui dan merupakan cara mudah untuk mengkompromikan sistem tanpa harus melawan mekanisme keamanan mutakhir, yang mungkin ada untuk melindungi versi API terbaru.	Dokumentasi yang sudah ketinggalan zaman membuatnya lebih sulit untuk menemukan dan/atau memperbaiki kerentanan. Kurangnya inventarisasi aset dan strategi pensiun menyebabkan menjalankan sistem yang tidak diperbarui, yang mengakibatkan kebocoran data sensitif. Umum ditemukan host API yang terpapar secara tidak perlu karena konsep modern seperti mikroservis, yang memudahkan aplikasi untuk diterapkan dan independen (misalnya, komputasi cloud, k8s).	Penyerang dapat memperoleh akses ke data sensitif, atau bahkan mengambil alih server melalui versi API lama yang tidak diperbarui yang terhubung ke basis data yang sama.

Apakah API Rentan?

API mungkin rentan jika:

- Tujuan dari host API tidak jelas, dan tidak ada jawaban eksplisit untuk pertanyaan berikut:
  - Lingkungan apa API berjalan (misalnya, produksi, staging, pengujian, pengembangan)?
  - Siapa yang seharusnya memiliki akses jaringan ke API (misalnya, publik, internal, mitra)?
  - Versi API apa yang berjalan?
  - Data apa yang dikumpulkan dan diproses oleh API (misalnya, PII)?
  - Bagaimana aliran datanya?
- Tidak ada dokumentasi, atau dokumentasi yang ada tidak diperbarui.
- Tidak ada rencana pensiun untuk setiap versi API.
- Inventarisasi host hilang atau ketinggalan zaman.
- Inventarisasi layanan terintegrasi, baik pihak pertama maupun ketiga, hilang atau ketinggalan zaman.

- Versi API lama atau sebelumnya berjalan tanpa patch.

Skenario Serangan Contoh

Skenario #1

Setelah merancang ulang aplikasi mereka, layanan pencarian lokal meninggalkan versi API lama (api.someservice.com/v1) berjalan, tidak dilindungi, dan dengan akses ke basis data pengguna. Saat menargetkan salah satu aplikasi rilis terbaru, seorang penyerang menemukan alamat API (api.someservice.com/v2). Mengganti v2 dengan v1 di URL memberi penyerang akses ke API lama, tidak dilindungi, yang memaparkan informasi identifikasi pribadi (PII) lebih dari 100 juta pengguna.

Skenario #2

Sebuah jaringan sosial menerapkan mekanisme pembatasan laju yang memblokir penyerang dari menggunakan brute-force untuk menebak token reset kata sandi. Mekanisme ini tidak diimplementasikan sebagai bagian dari kode API itu sendiri, tetapi dalam komponen terpisah antara klien dan API resmi (www.socialnetwork.com). Seorang peneliti menemukan host API beta (www.mbasic.beta.socialnetwork.com) yang menjalankan API yang sama, termasuk mekanisme reset kata sandi, tetapi mekanisme pembatasan laju tidak diterapkan. Peneliti dapat mereset kata sandi pengguna mana pun dengan menggunakan brute-force sederhana untuk menebak token 6 digit.

Cara Mencegah

- Inventarisasi semua host API dan dokumentasikan aspek penting dari masing-masing, berfokus pada lingkungan API (misalnya, produksi, staging, pengujian, pengembangan), siapa yang seharusnya memiliki akses jaringan ke host (misalnya, publik, internal, mitra) dan versi API.
- Inventarisasi layanan terintegrasi dan dokumentasikan aspek penting seperti peran mereka dalam sistem, data apa yang dipertukarkan (aliran data), dan sensitivitasnya.
- Dokumentasikan semua aspek API Anda seperti otentikasi, kesalahan, pengalihan, pembatasan laju, kebijakan berbagi sumber daya lintas asal (CORS) dan endpoint, termasuk parameter, permintaan, dan respons mereka.
- Hasilkan dokumentasi secara otomatis dengan mengadopsi standar terbuka. Sertakan pembangunan dokumentasi dalam pipeline CI/CD Anda.
- Buat dokumentasi API tersedia untuk mereka yang berwenang menggunakan API.
- Gunakan langkah-langkah perlindungan eksternal seperti firewall keamanan API untuk semua versi terekspos API Anda, bukan hanya untuk versi produksi saat ini.

- Hindari menggunakan data produksi dengan penerapan API non-produksi. Jika ini tidak dapat dihindari, endpoint ini harus mendapatkan perlakuan keamanan yang sama dengan produksi.
- Ketika versi API yang lebih baru mencakup peningkatan keamanan, lakukan analisis risiko untuk membuat keputusan tindakan mitigasi yang diperlukan untuk versi yang lebih tua: misalnya, apakah mungkin menerapkan peningkatan tanpa merusak kompatibilitas API atau Anda perlu menarik versi yang lebih tua dengan cepat dan memaksa semua klien beralih ke versi terbaru.

Referensi

Eksternal

- CWE-1059: Dokumentasi yang Tidak Lengkap
- Inisiatif OpenAPI

API10:2019 Pencatatan & Pemantauan yang Tidak Memadai

Agen Ancaman/Vektor Serangan	Kelemahan Keamanan	Dampak
Khusus API: Eksploitasi 2	Prevalensi **3** : Deteksi **1**	Teknis **2** : Spesifik Bisnis
Penyerang memanfaatkan kurangnya pencatatan dan pemantauan untuk menyalahgunakan sistem tanpa disadari.	Tanpa pencatatan dan pemantauan, atau dengan pencatatan dan pemantauan yang tidak memadai, hampir mustahil untuk melacak kegiatan mencurigakan dan menanggapinya tepat waktu.	Tanpa visibilitas atas kegiatan berbahaya yang sedang berlangsung, penyerang memiliki banyak waktu untuk sepenuhnya mengkompromikan sistem.

Apakah API Rentan?

API rentan jika:

- Tidak menghasilkan log apa pun, level pencatatan tidak disetel dengan benar, atau pesan log tidak menyertakan detail yang cukup.
- Integritas log tidak dijamin (misalnya, Log Injection).
- Log tidak dipantau secara terus menerus.
- Infrastruktur API tidak dipantau secara terus menerus.

Skenario Serangan Contoh

Skenario #1

Kunci akses administratif API bocor di repositori publik. Pemilik repositori diberi tahu melalui email tentang kebocoran potensial, tetapi membutuhkan waktu lebih dari 48 jam untuk menindaklanjuti insiden, dan paparan kunci akses mungkin telah mengizinkan akses ke data sensitif. Karena pencatatan yang tidak memadai, perusahaan tidak dapat menilai data apa yang diakses oleh aktor berbahaya.

Skenario #2

Platform berbagi video terkena serangan "skala besar" stuffing kredensial. Meskipun login gagal dicatat, tidak ada peringatan yang dipicu selama rentang waktu serangan. Sebagai reaksi atas keluhan pengguna, log API dianalisis dan serangan terdeteksi. Perusahaan

harus membuat pengumuman publik yang meminta pengguna mereset kata sandi mereka, dan melaporkan insiden kepada otoritas peraturan.

Cara Mencegah

- Catat semua upaya otentikasi gagal, akses yang ditolak, dan kesalahan validasi input.
- Log harus ditulis menggunakan format yang sesuai untuk dikonsumsi oleh solusi manajemen log, dan harus mencakup detail yang cukup untuk mengidentifikasi pelaku jahat.
- Log harus ditangani sebagai data sensitif, dan integritasnya harus dijamin saat diam dan dalam transit.
- Konfigurasikan sistem pemantauan untuk secara terus menerus memantau infrastruktur, jaringan, dan fungsi API.
- Gunakan sistem Manajemen Informasi dan Keamanan (SIEM) untuk menggabungkan dan mengelola log dari semua komponen tumpukan API dan host.
- Konfigurasikan dashboard dan peringatan kustom, memungkinkan kegiatan mencurigakan terdeteksi dan direspon lebih awal.

Referensi

OWASP

- OWASP Logging Cheat Sheet
- OWASP Proactive Controls: Implement Logging and Intrusion Detection
- OWASP Application Security Verification Standard: V7: Error Handling and Logging Verification Requirements

Eksternal

- CWE-223: Omission of Security-relevant Information
- CWE-778: Insufficient Logging

Apa Selanjutnya untuk Pengembang

Tugas untuk membuat dan memelihara perangkat lunak yang aman, atau memperbaiki perangkat lunak yang ada, dapat sulit. API tidak berbeda.

Kami yakin bahwa pendidikan dan kesadaran adalah faktor kunci untuk menulis perangkat lunak yang aman. Semua hal lain yang diperlukan untuk mencapai tujuan, bergantung pada **membangun dan menggunakan proses keamanan yang dapat diulang dan kontrol keamanan standar**.

OWASP memiliki berbagai sumber daya gratis dan terbuka untuk mengatasi masalah keamanan sejak awal proyek. Silakan kunjungi halaman Proyek OWASP untuk daftar lengkap proyek yang tersedia.

Pendidikan	Anda dapat mulai membaca materi Proyek Pendidikan OWASP sesuai dengan profesi dan minat Anda. Untuk pembelajaran hands-on, kami menambahkan **crAPI** - **C**ompletely **R**idiculous **API** dalam roadmap kami. Sementara itu, Anda dapat berlatih WebAppSec menggunakan Modul Pixi DevSlop OWASP, layanan WebApp dan API rentan yang bertujuan untuk mengajari pengguna cara menguji aplikasi web dan API modern untuk masalah keamanan, dan cara menulis API yang lebih aman di masa depan. Anda juga dapat menghadiri sesi pelatihan Konferensi OWASP AppSec, atau bergabung dengan chapter lokal Anda.
Persyaratan Keamanan	Keamanan harus menjadi bagian dari setiap proyek sejak awal. Saat melakukan elicitation persyaratan, penting untuk mendefinisikan apa artinya "aman" untuk proyek tersebut. OWASP merekomendasikan Anda menggunakan OWASP Application Security Verification Standard (ASVS) sebagai panduan untuk menetapkan persyaratan keamanan. Jika Anda outsourcing, pertimbangkan OWASP Secure Software Contract Annex, yang harus disesuaikan sesuai hukum dan peraturan setempat.
Arsitektur Keamanan	Keamanan harus tetap menjadi perhatian selama semua tahapan proyek. OWASP Prevention Cheat Sheets merupakan titik awal yang baik untuk panduan tentang cara merancang keamanan selama fase arsitektur. Di antara banyak lainnya, Anda akan menemukan REST Security Cheat Sheet dan REST Assessment Cheat Sheet.

Kontrol Keamanan Standar	Mengadopsi Kontrol Keamanan Standar mengurangi risiko memperkenalkan kelemahan keamanan saat menulis logika Anda sendiri. Meskipun fakta banyak kerangka kerja modern sekarang datang dengan kontrol efektif standar bawaan, OWASP Proactive Controls memberi Anda gambaran yang baik tentang kontrol keamanan apa yang harus Anda cari untuk dimasukkan dalam proyek Anda. OWASP juga menyediakan beberapa pustaka dan alat yang mungkin Anda anggap berharga, seperti kontrol validasi.
Siklus Hidup Pengembangan Perangkat Lunak yang Aman	Anda dapat menggunakan OWASP Software Assurance Maturity Model (SAMM) untuk meningkatkan proses saat membangun API. Beberapa proyek OWASP lainnya tersedia untuk membantu Anda selama fase pengembangan API yang berbeda misalnya, OWASP Code Review Project.

Apa Selanjutnya untuk DevSecOps

Karena pentingnya dalam arsitektur aplikasi modern, membangun API yang aman sangat penting. Keamanan tidak boleh diabaikan, dan itu harus menjadi bagian dari seluruh siklus pengembangan. Pemindaian dan penetration testing setahun sekali tidak lagi cukup.

DevSecOps harus bergabung dengan upaya pengembangan, memfasilitasi pengujian keamanan yang berkelanjutan di seluruh siklus pengembangan perangkat lunak. Tujuan mereka adalah untuk meningkatkan pipeline pengembangan dengan otomatisasi keamanan, dan tanpa berdampak pada kecepatan pengembangan.

Jika ragu, tetap terinformasi, dan tinjau, DevSecOps Manifesto sering.

Pahami Model Ancaman	Prioritas pengujian berasal dari model ancaman. Jika Anda tidak memilikinya, pertimbangkan menggunakan OWASP Application Security Verification Standard (ASVS), dan OWASP Testing Guide sebagai input. Melibatkan tim pengembangan dapat membantu membuat mereka lebih sadar keamanan.
Pahami SDLC	Bergabung dengan tim pengembangan untuk lebih memahami Siklus Hidup Pengembangan Perangkat Lunak. Kontribusi Anda pada pengujian keamanan berkelanjutan harus kompatibel dengan orang, proses, dan alat. Semua orang harus setuju dengan prosesnya, sehingga tidak ada gesekan atau perlawanan yang tidak perlu.
Strategi Pengujian	Karena pekerjaan Anda tidak boleh berdampak pada kecepatan pengembangan, Anda harus bijaksana memilih teknik terbaik (sederhana, tercepat, paling akurat) untuk memverifikasi persyaratan keamanan. OWASP Security Knowledge Framework dan OWASP Application Security Verification Standard dapat menjadi sumber persyaratan keamanan fungsional dan non-fungsional yang hebat. Ada sumber lain yang hebat untuk proyek dan alat serupa dengan yang ditawarkan oleh komunitas DevSecOps.
Meraih Cakupan dan Akurasi	Anda adalah jembatan antara tim pengembang dan operasi. Untuk mencapai cakupan, Anda tidak hanya harus fokus pada fungsionalitas, tetapi juga orkestrasi. Bekerja dekat dengan tim pengembangan dan operasi dari awal sehingga Anda dapat mengoptimalkan waktu dan upaya Anda. Anda harus menargetkan keadaan di mana keamanan esensial diverifikasi secara berkelanjutan.

Komunikasikan Temuan dengan Jelas	Berikan nilai dengan sedikit atau tanpa gesekan. Kirim temuan tepat waktu, dalam alat yang digunakan tim pengembangan (bukan file PDF). Bergabung dengan tim pengembangan untuk menangani temuan. Manfaatkan kesempatan untuk mendidik mereka, menjelaskan kelemahan dan bagaimana dapat disalahgunakan, termasuk skenario serangan untuk membuatnya nyata.

Metodologi dan Data

Ikhtisar

Karena industri AppSec belum secara khusus difokuskan pada arsitektur aplikasi paling mutakhir, di mana API memainkan peran penting, menyusun daftar sepuluh risiko keamanan API paling kritis, berdasarkan panggilan data publik, akan menjadi tugas yang sulit. Meskipun tidak ada panggilan data publik, daftar Top 10 yang dihasilkan masih didasarkan pada data yang tersedia untuk publik, kontribusi pakar keamanan, dan diskusi terbuka dengan komunitas keamanan.

Metodologi

Pada fase pertama, data publik tentang insiden keamanan API dikumpulkan, ditinjau, dan dikategorikan oleh sekelompok pakar keamanan. Data tersebut dikumpulkan dari platform bug bounty dan basis data kerentanan, dalam rentang waktu satu tahun. Itu digunakan untuk tujuan statistik.

Pada fase berikutnya, praktisi keamanan dengan pengalaman penetration testing diminta untuk menyusun daftar Top 10 mereka sendiri.

OWASP Risk Rating Methodology digunakan untuk melakukan Analisis Risiko. Skor didiskusikan dan ditinjau di antara praktisi keamanan. Untuk pertimbangan mengenai hal ini, harap merujuk pada bagian API Security Risks.

Draf pertama OWASP API Security Top 10 2019 dihasilkan dari konsensus antara hasil statistik dari fase satu, dan daftar praktisi keamanan. Draf ini kemudian diserahkan untuk penghargaan dan tinjauan oleh kelompok praktisi keamanan lainnya, dengan pengalaman yang relevan di bidang keamanan API.

OWASP API Security Top 10 2019 pertama kali dipresentasikan dalam acara OWASP Global AppSec Tel Aviv (Mei 2019). Sejak saat itu, telah tersedia di GitHub untuk diskusi publik dan kontribusi.

Daftar kontributor tersedia di bagian Penghargaan.

Ucapan Terima Kasih

Ucapan Terima Kasih kepada Kontributor

Kami ingin berterima kasih kepada kontributor berikut yang berkontribusi secara publik di GitHub atau melalui cara lain:

- 007divyachawla
- Abid Khan
- Adam Fisher
- anotherik
- bkimminich
- caseysoftware
- Chris Westphal
- dsopas
- DSotnikov
- emilva
- ErezYalon
- faizzaidi
- flascelles
- Guillaume Benats
- IgorSasovets
- Inonshk
- JonnySchnittger
- jmanico
- jmdx
- Keith Casey
- kozmic
- LauraRosePorter
- Matthieu Estrade
- nathanawmk
- PauloASilva
- pentagramz
- philippederyck
- pleothaud
- r00ter
- Raj kumar
- Sagar Popat
- Stephen Gates
- thomaskonrad
- xycloops123