Muhamad Hidayat Follow
Sep 29, 2022 · 5 min read

OWASP API Security (Offensive Prespective) : Broken Function Level Authorization #5

Broken Gears Foto Stok, Potret & Gambar Bebas Royalti — iStock (istockphoto.com)

Assalamualaikum Wr. Wb

Pada artikel kali ini saya mengunakan lab yang di develop oleh Insider PHd, untuk nantinya melakukan demostrasi terkait Broken Function Level Authorization.

Docker Version: busk3r/genericuniversity — Docker Image | Docker Hub
Kali Linux: Generic-University/KaliSetup.md at master · InsiderPhD/Generic-University · GitHub

Apa itu Otorisasi/Authorization?

Ialah sebuah **hak atau wewenang** yang terdapat pada individu, dalam konteks aplikasi adalah ketika sebuah user memiliki sebuah hak akses atau kewenangan terhadap resource tertentu.

Contoh singkat Otorisasi terkait **BFLA,** Ayah membuat peraturan bahwa anak-anak hanya boleh bermain *game console* pada *Weekend* saja (Sabtu-Minggu), Diluar hari tersebut sang ayah menyembunyikan *game console* tersebut dalam lemari yang terdapat didalam kamar ayah.

Disini bisa kita simpulkan bahwa sang ayah memiliki otorisasi atau hak untuk menyembunyikan *console* tersebut, Dalam konteks aplikasi **ayah** adalah sebuah peran atau role pada sebuah user yang memiliki akses penuh terhadap *resource* tertentu (dalam hal ini *Game Console*).

Karena sang anak tidak sabar, maka sang anak mencoba mencari akal bagaimana untuk dapat mendapatkan game console tersebut dan tetap bisa memainkanya pada *Weekdays,* Dengan cara masuk ke kamar ayahnya tanpa ketahuan, Dalam sebuah aplikasi aktivitas anak tersebut dinamakan **Bypassing Access Control.** Yaitu aktivitas penembusan sebuah sistem kontrol yang sudah diberi keamanan penuh.

Singkat cerita, sang anak bisa mendapatkan game console tersebut dengan cara membohongi ibunya dengan berbohong *"ibu, barusan aku telfon ayah kata ayah aku boleh main game console hari ini, karna hari ini nilai ujian ku bagus."* , dan ibunya pun mengambil console tersebut dan memberikanya kepada anaknya.

Pada konteks aplikasi, sang ayah (Admin Role) yang memiliki hak penuh terhadap suatu resource (Game console) telah melakukan limitasi akses terhadap anak-anaknya (Low level Role), tidak disangka sang anak bisa mengakali sistem tersebut (Restricted Resource) dengan cara membohongi atas nama ayahnya (Impersonating).

Untuk penjelasan lebih detail terkait otorisasi terdapat pada article terkait OWASP API Security (Offensive Prespective) : Broken Object Level Authorization #1 | by Muhamad Hidayat | Medium.

Perbedaan Antara Broken Object Level Authorization & Broken Function Level Authorization?

Perbedaan sederhananya adalah **BOLA** lebih kearah gagalnya keamanan mengatur sebuah otorisasi pada resource tiap-tiap object / individu, dengan kata lain tiap-tiap object/individu dapat mengakses satu sama lain tanpa ada nya limitasi.

Contoh seperti kasus IDOR (Insecure Direct Object Refference), yaitu ketika user dengan id 10001 dapat melakukan akses pada resource 10002 dan seterusnya, otorisasi level berada pada object / individu itu sendiri.

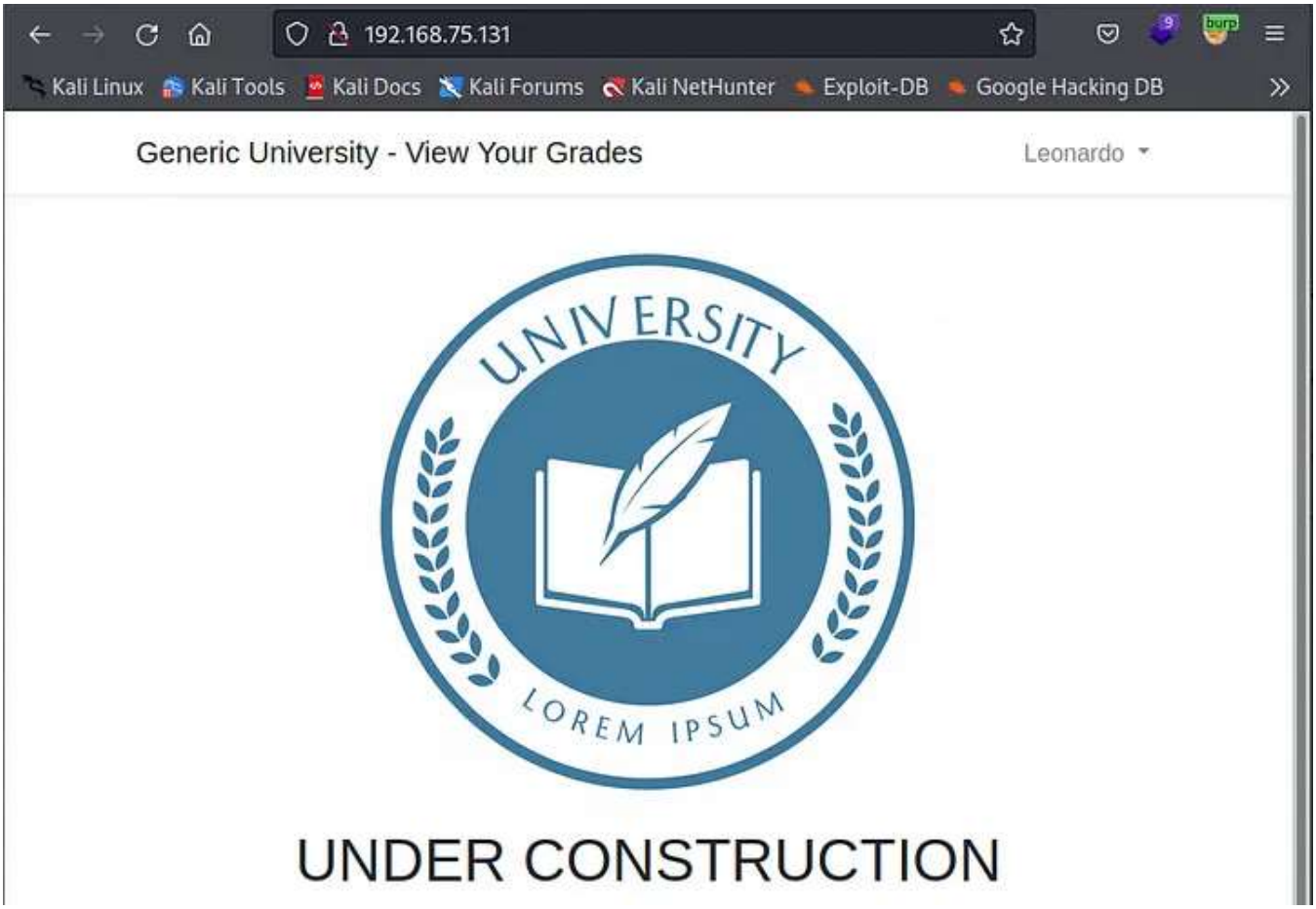
Berbeda dengan **BFLA** lebih mengarah pada gagalnya keamanan mengatur sebuah otorisasi resource pada perbedaan level fungsi atau hirarki.

Contoh seperti kasus **Broken Access Control,** yaitu ketika user dengan low level dapat mengakses resource yang hanya dapat diakses oleh user high level (Administrator).

Hal diatas telah di jelaskan detail dengan contoh kasus **Access Data Object & Access Function Object** pada artikel:

OWASP API Security (Offensive Prespective) : Broken Object Level Authorization #1 | by Muhamad Hidayat | Medium

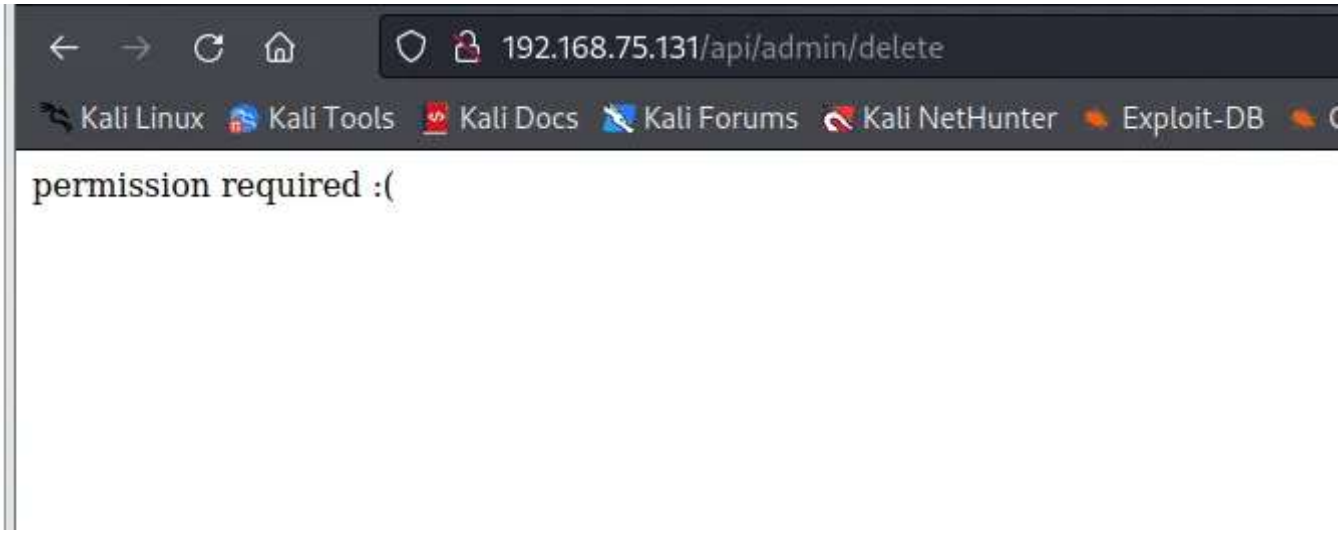
Praktikum dengan Generic University



Homepage Generic University

Kita telah mengakses web generic university dengan user mahasiswa yaitu **Leonardo,** singkat cerita leonardo berencana untuk melakukan peretasan terhadap situs kampusnya sendiri.

Leo berencana untuk melakukan drop DB atau penghapusan full database pada situs kampusnya, namun yang terjadi..



Gagal akses fungsi karena role tidak sesuai.

Berarti leo memiliki sebuah goals awal yaitu untuk mendapatkan dahulu role atau permission yang tepat agar dapat mengakses fungsi /delete pada website tersebut yaitu admin.


```
{
  "id":20,
  "name":"IT Florian Wintheiser",
  "email":"victoria.spencer@hotmail.com",
  "email_verified_at":null,
  "created_at":"2021-04-18T10:41:26.000000Z",
  "updated_at":"2021-04-18T10:41:26.000000Z",
  "role_id":1
},
{
  "id":21,
  "name":"Dr Gwendolyn Block",
  "email":"kieran.koepp@mayert.org",
  "email_verified_at":null,
  "created_at":"2021-04-18T10:41:26.000000Z",
  "updated_at":"2021-04-18T10:41:26.000000Z",
  "role_id":3
},
{
  "id":22,
  "name":"Leonardo",
  "email":"leo@sgilabs.co.id",
  "email_verified_at":null,
  "created_at":"2022-09-29T14:57:04.000000Z",
  "updated_at":"2022-09-29T14:57:04.000000Z",
  "role_id":2
}
```


list user pada website kampus

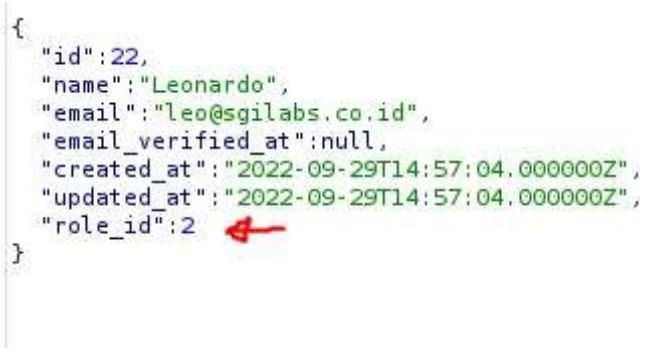
Terlihat terdapat 3 object json berisi 3 buah data dengan role_id yang berbeda yaitu 1,3,2 leo mendapatkan informasi yang cukup terkait role_id yang ada pada website tersebut yang nantinya akan bisa di kombinasikan dengan serangan yang lain.


```
[
  {
    "id":1,
    "name":"Admin",
    "created_at":"2021-04-18T09:34:27.000000Z",
    "updated_at":"2021-04-18T09:34:27.000000Z"
  },
  {
    "id":2,
    "name":"Student",
    "created_at":"2021-04-18T09:34:27.000000Z",
    "updated_at":"2021-04-18T09:34:27.000000Z"
  },
  {
    "id":3,
    "name":"Teacher",
    "created_at":"2021-04-18T09:34:27.000000Z",
    "updated_at":"2021-04-18T09:34:27.000000Z"
  },
```


Arti dari tiap nomor role_id 1,2,3

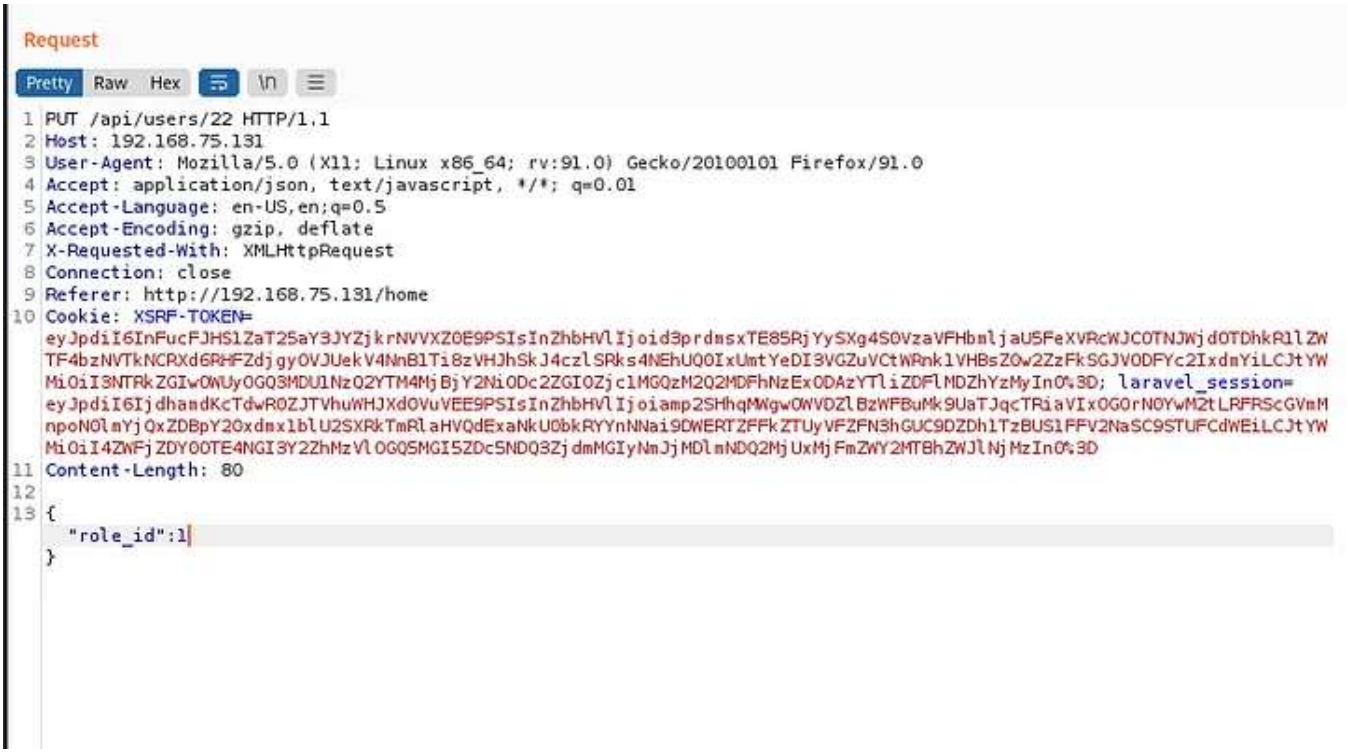
Singkat cerita Leo menemukan sebuah bug yaitu dapat melakukan pengubahan data pada user mana pun. dengan cara mengubah http method request GET menjadi PUT.

Leonardo mencoba melakukan komparasi untuk mengetahui apakah exploitasi tersebut berhasil, apakah data yang ingin diubah benar-benar berubah?



Kondisi role_id sebelum diubah

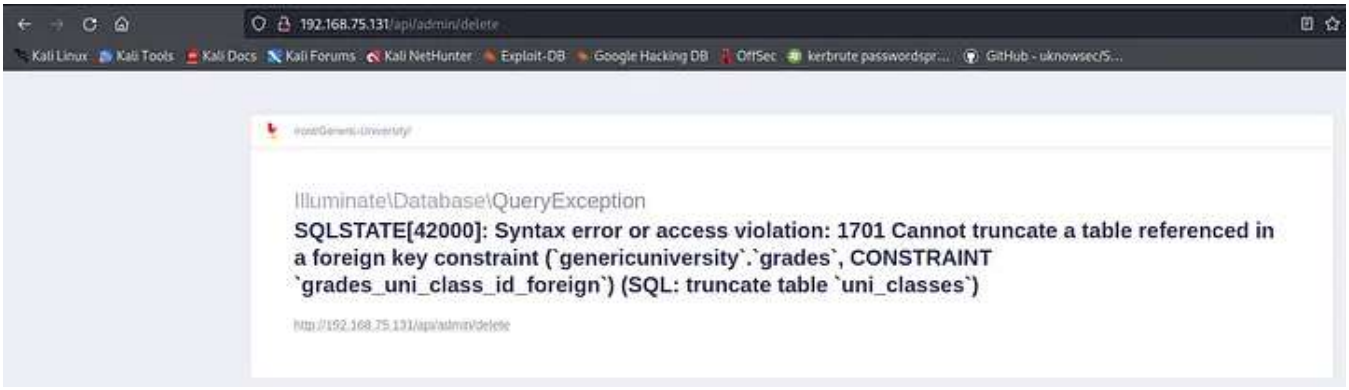
Lalu leo mengubah data tersebut dengan metode put dan memberikan request role_id=1 untuk mengubah spesifik data yang ingin diubah.



request mengubah data leonardo

Nampaknya berhasil, leo telah mengubah user nya sendiri menjadi role admin.

Kondisi role_id setelah diubah



Leonardo berhasil melakukan bypass terhadap fungsi delet pada admin

Sayang, walau leonardo telah melakukan privilage escalation untuk mengakses fungsi yang hanya bisa dijalankan oleh high level (admin) namun database melarang akses penghapusan data tersebut.

Aktivitas yang dilakukan leonardo adalah melakukan tindakan yang hanya dapat diakases oleh entitas lebih tinggi namun leonardo dapat melakukanya dengan user low level lalu mencoba melakukan eskalasi untuk mendapatkan akses fungsi tersebut.

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.

Wassalam.

Broken Access Control Owasp Top 10 Api Security Authorization Hacking