How to Protect Firebase Http Cloud Functions

nschairer

Aug 4, 2018 · 4 min read ★

When you create a Http Cloud Function using Firebase, you are creating a real url that anyone can put into their browser, postman, etc. and run a request. This is very inconvenient because every time that url is hit with a request your function is going to run. There is a very simple way to fix this, it is done by setting environment variables for your functions using the Firebase CLI tool. If you are not sure how to setup your Firebase cloud functions from scratch, refer to my previous tutorial here.

Creating your custom key

The first step in protecting your function is choosing a key. Really you can make it whatever you would like, but realistically, it is safer to create a key using a third party like crypto which will generate a random token for you. In your terminal type the following:

```
$ node
$ crypto.randomBytes(20).toString('hex')
```

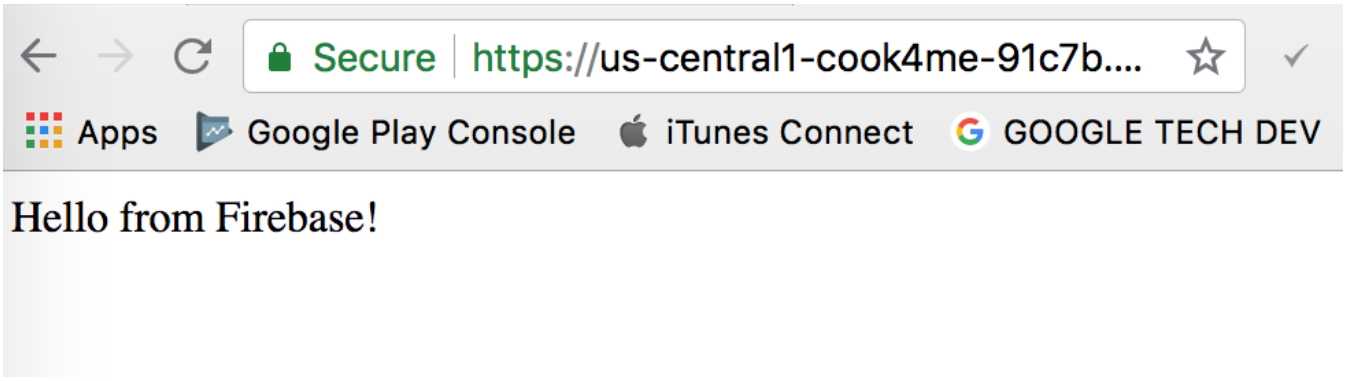
The output should look something like this

```
'3cd0955628f6def1263c70c83b6e88e47ef66c96'
```

Congratulations! You have created a key! **(DON'T USE THIS ONE^)**

Setting your custom key

For the purpose of this tutorial we will refer to the function we deployed in the previous



Response from helloWorld function

In order to protect this function with our custom key, we need to set an environment variable. It is very simple, make sure you are in the **correct directory in your terminal**, and type the following:

```
firebase functions:config:set helloworld.key = "<YOUR CUSTOM KEY>"
```

A couple things to note, the part that says “helloworld.key”, is not actually related to your function in any way. If you tried to use helloWorld, Firebase CLI would actually throw an error for including a capital letter in the name. All this is, is an environment variable, meaning that we can retrieve it from within our cloud function at any time. Also, if you set a key and forgot it, you can simply run:

```
firebase functions:config:get helloworld.key
```

This will return the key you just set. Awesome! We are ready to do some coding to make some use of this environment variable.

Using your key

Go to your index.js file and go to the function that you would like to protect with your newly created key. Within the function, type:

```
const key = functions.config().helloworld.key;
```

This accesses the key that we just set. Next we need to retrieve the key from the incoming request, to do this we need to choose the name of the header that the key will be in. For this example we will use ‘auth’. Add the following line to your program:

```
const req_key = request.get('auth');
```

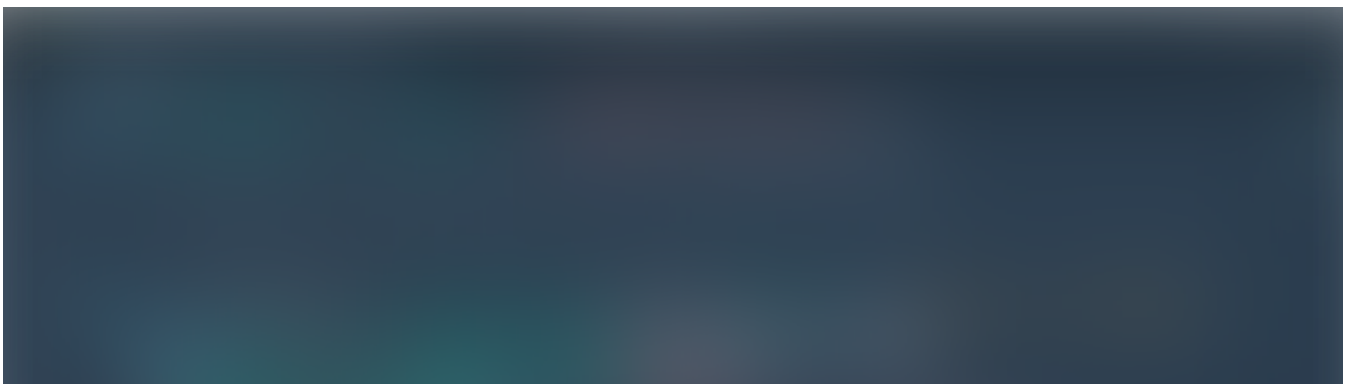
Finally, we need to compare the two keys. We will make sure the key that we set matches the key sent with the request to our function. Set a simple conditional statement like this:

```
if (key === req_key) {
     response.status(200).send('Hello from Firebase!');
} else {
     response.status(400).send('Error 400: Bad key');
}
```

All we did here was compare our key with the one used to make the request. Also, remember to always send back a request even if it is a bad key, otherwise the function will timeout. Redeploy your function by entering the following in the terminal:

```
firebase deploy --only functions:helloWorld
```

If you had any errors, here is what the code should look like when you open your index.js file:



Cloud Function Completed

Finally, lets see if it worked! Enter your function url in your browser, we should get back ‘Error 400: Bad key’.

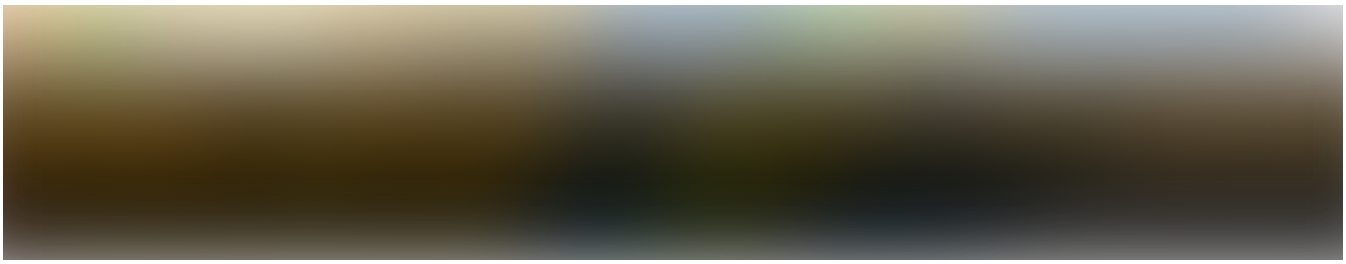
Bad Request

As you can see we have now protected our function from anyone without the proper key! Any code you run in that else statement will now run if there is a bad request.

You may be wondering how we verify that our key works? All we have done is shown that we cannot use our function anymore? It is simple, all we have to do is create a request using CURL… let me show you. Type the following into your terminal:

```
curl --header "auth: < YOUR KEY >" <YOUR FUNCTION URL>
```

I know the YOUR KEY and YOUR FUNCTION URL might make this confusing so here is a picture of what a proper CURL request looks like in a terminal:



Curl Request

Succesful Response

Congratulations!

You can now protect your http functions with this method! It is a pretty simple way to add some extra security to your cloud functions. If you have any questions don't hesitate to ask in the comments! Also, feel free to check out my previous post if you are still unsure of how to setup your Cloud Functions.

Nodejs Firebase Firebase Cloud Functions Learn To Code Programming