Apigee Lab 6a: Masking Sensitive Data

experiment Lab schedule 1 hour 30 minutes universal_currency_alt No cost

show_chart Introductory

Overview

In this lab, you use debug masks and private variables to protect sensitive data from being viewed in the debug tool.

Objectives

In this lab, you learn how to perform the following tasks:

- Use private variables in API proxies.
- Create debug masks using the Apigee API.

Setup

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Sign in to Qwiklabs using an **incognito window**.

2. Note the lab's access time (for example, `1:15:00`), and make sure you can finish within that time.
   There is no pause feature. You can restart if needed, but you have to start at the beginning.

3. When ready, click **Start lab**.

4. Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.

5. Click **Open Google Console**.

6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.
   If you use other credentials, you'll receive errors or **incur charges**.

7. Accept the terms and skip the recovery resource page.

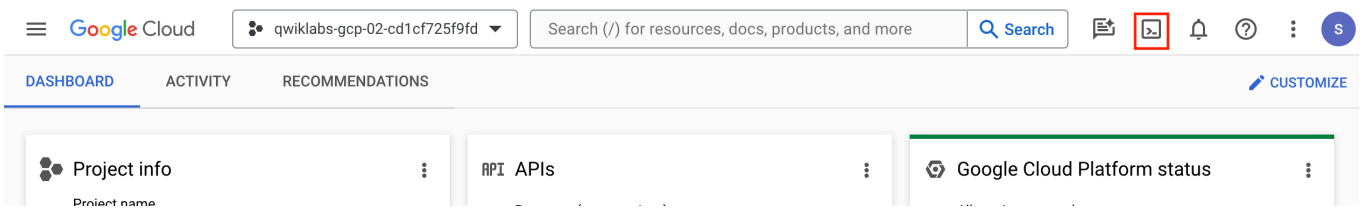
Note: Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

Activate Google Cloud Shell

Google Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud.

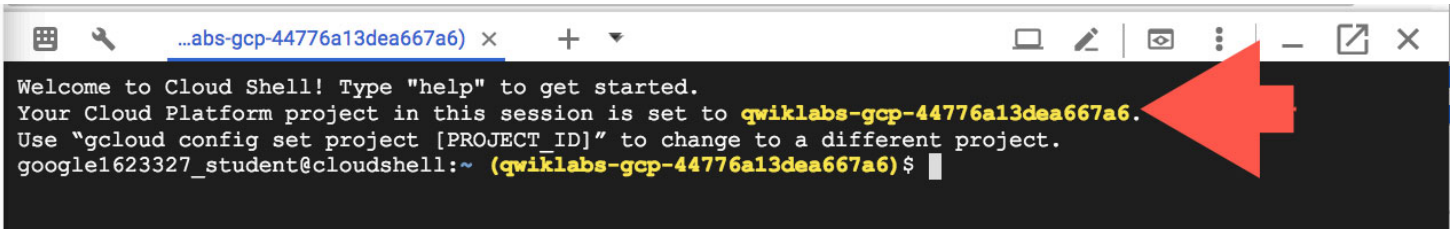
Google Cloud Shell provides command-line access to your Google Cloud resources.

1. In Cloud console, on the top right toolbar, click the Open Cloud Shell button.



2. Click **Continue**.

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



gcloud is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

- You can list the active account name with this command:

```
gcloud auth list
```

content_c

Output:

```
Credentialed accounts:
 - @.com (active)
```

Example output:

```
Credentialed accounts:
 - google1623327_student@qwiklabs.net
```

- You can list the project ID with this command:

```
gcloud config list project
```

content_c

Output:

```
[core]
project =
```

Example output:

```
[core]
project = qwiklabs-gcp-44776a13dea667a6
```

Note: Full documentation of **gcloud** is available in the gcloud CLI overview guide .

Task 1. Create a new proxy

In this task, you create a new API proxy.

1. In the Google Cloud console, on the **Navigation menu** (☰), select **Integration Services > Apigee > Proxy Development > API proxies**.

2. To start the proxy wizard, click **+Create**.

3. Leave **Proxy template** unchanged.

4. Specify the following settings:

Property	Value
Proxy Name	**lab6a-v1**
Base path	**/lab6a/v1**
Target (Existing API)	**https://httpbin.org/anything**

The **httpbin.org/anything** API returns detailed information about the API request it was sent.

Note: Confirm that you are using "/lab6a/v1" for the base path, and not "/lab6a-v1".

5. Click **Create**.

6. Click the **Develop** tab.

Task 2. Use a private variable in the proxy to hide sensitive data

In this task, you test the difference between private variables and non-private variables when viewed in the debug tool.

Add an ExtractVariables policy

1. Click **Proxy endpoints > default > PreFlow**.
2. On the **Request PreFlow**, click **Add Policy Step (+)**.
3. In the **Add policy step** pane, select **Create new policy**, and then select **Mediation > Extract Variables**.
4. Specify the following values:

Property	**Value**
Name	**EV-QueryParamTest**
Display name	**EV-QueryParamTest**

5. Click **Add**.
6. Click **Policies > EV-QueryParamTest**.
7. Replace the policy's default configuration with:

```
<ExtractVariables continueOnError="false" enabled="true"
name="EV-QueryParamTest">
  <QueryParam name="user">
    <Pattern ignoreCase="true">{username}</Pattern>
  </QueryParam>
  <QueryParam name="pw">
    <Pattern ignoreCase="true">{password}</Pattern>
  </QueryParam>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <Source clearPayload="false">request</Source>
</ExtractVariables>
```

content_c

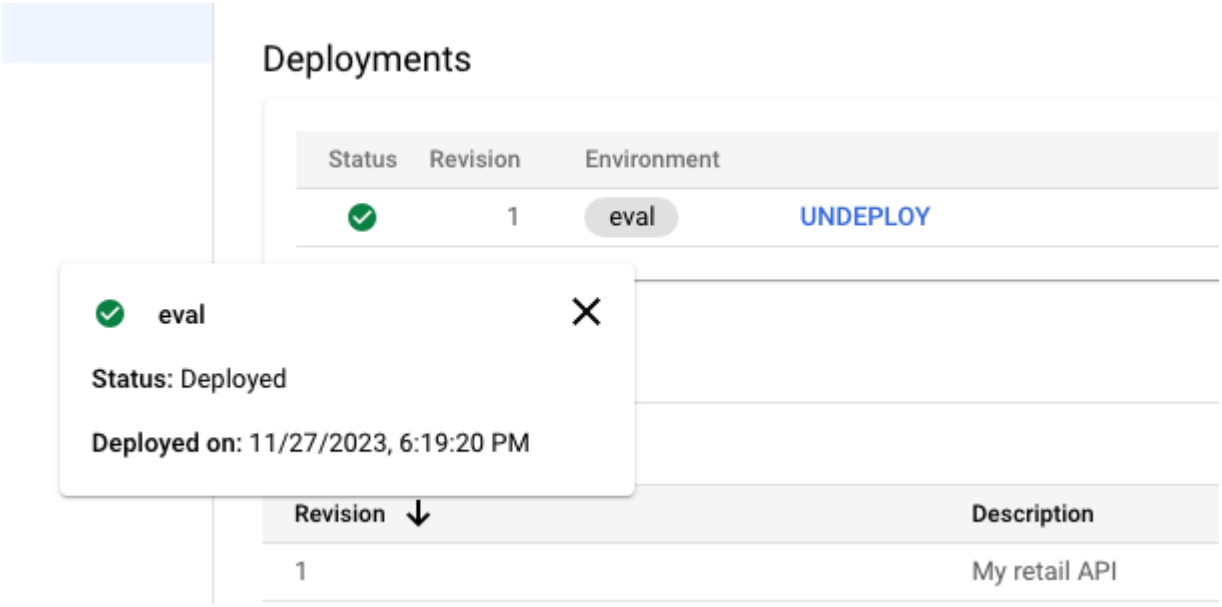
This configuration tells the proxy to look for a query parameter named **user** and copy its value to a variable named **username**, and to look for a query parameter named **pw** and copy it to a variable named **password**.

Note: You should not send sensitive data in the URL, because URLs are often logged in access logs. This example is being used for ease of testing.

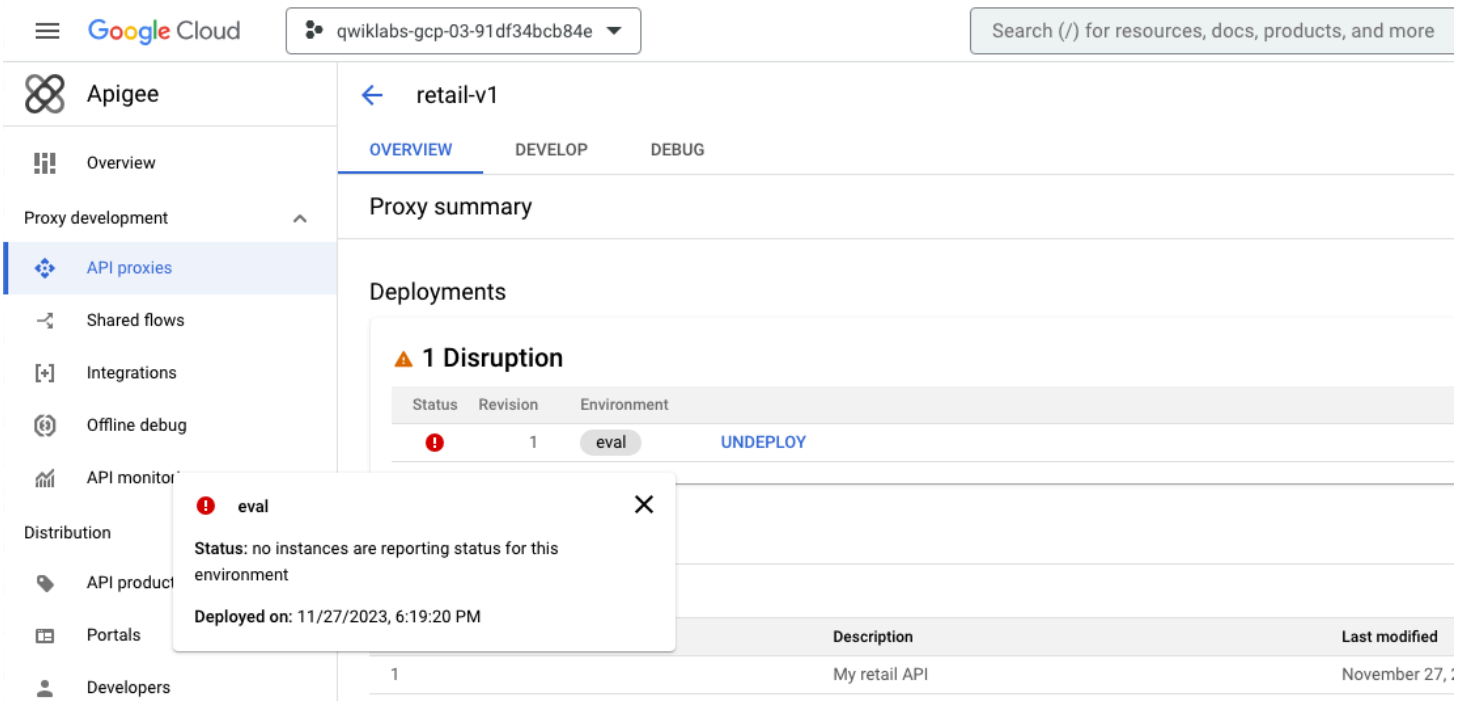
8. Click **Save**.
9. Click **Deploy**.
10. To specify that you want the new revision deployed to the eval environment, select **eval** as the **Environment**, and then click **Deploy**.
11. Click **Confirm**.

Check deployment status

A proxy that is deployed and ready to take traffic will show a green status.



When a proxy is marked as deployed but the runtime is not yet available and the environment is not yet attached, you may see a red warning sign. Hold the pointer over the **Status** icon to see the current status.

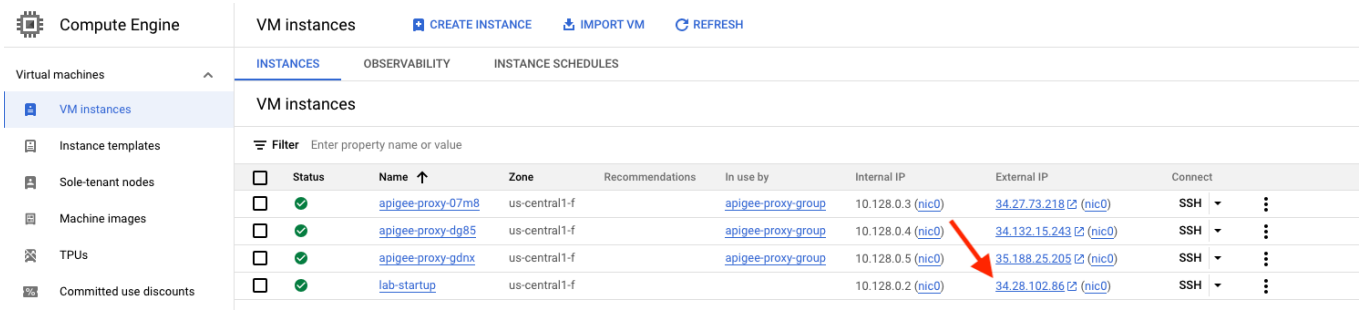


If the proxy is deployed and shows as green, your proxy is ready for API traffic. If your proxy is not deployed because there are no runtime pods, you can check the status of provisioning.

Check provisioning dashboard

1. In the Google Cloud Console, navigate to **Compute Engine > VM instances**.

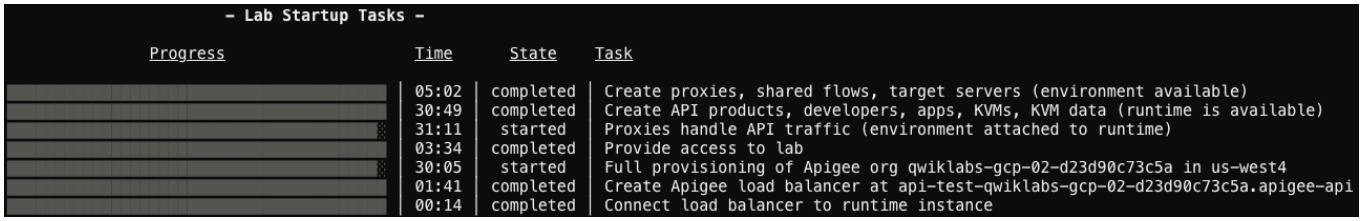
2. To open the Lab Startup Tasks dashboard, click on the **External IP** for the **lab-startup** VM.



3. If you see a redirect notice page, click the link to the external IP address.

A new browser window will open. Lab startup tasks are shown with their progress.

- *Create proxies, shared flows, target servers* should be complete when you first enter the lab, allowing you to use the Apigee console for tasks like proxy editing.
- *Create API products, developers, apps, KVMs, KVM data* indicates when the runtime is available and those assets may be saved.
- *Proxies handle API traffic* indicates when the eval environment has been attached to the runtime and the deployed proxies can take runtime traffic.



In this case, you need to wait for *Proxies handle API traffic* to complete.

While you are waiting

- Reference for the Apigee API: Apigee API reference
- Documentation on how to use private variables and debug masks: Masking and hiding data

Task 3. Test the API proxy without private variables

In this task, you will use the debug tool to verify that sensitive variables are visible.

Start a debug session

1. Click the **Debug** tab, and then click **Start Debug Session**.

2. In the **Start debug session** pane, on the Environment dropdown, select **eval**.

3. Click **Start**.

Test the API proxy

1. In Cloud Shell, send the following curl command:

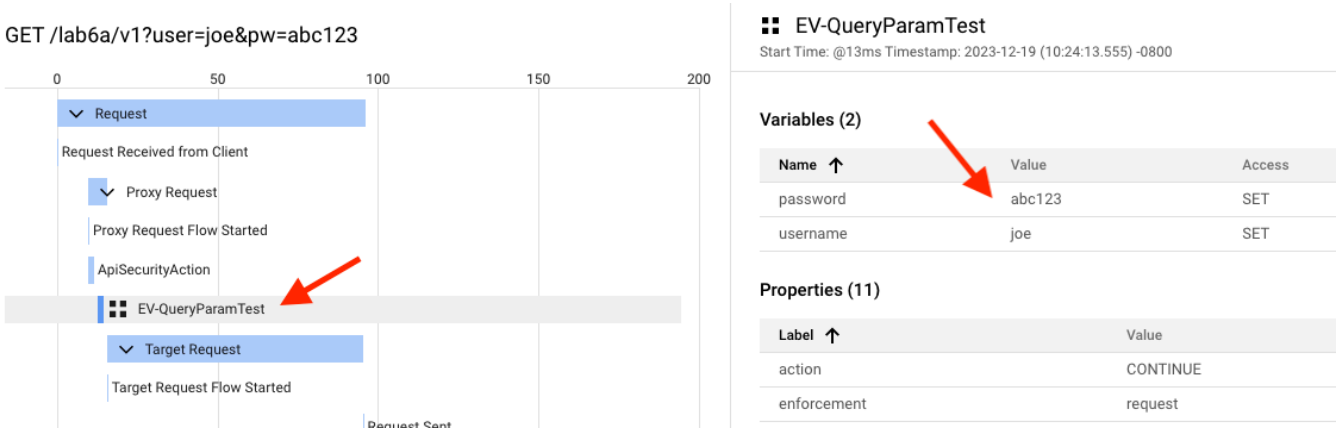
```
curl -X GET "https://api-test-${GOOGLE_CLOUD_PROJECT}.apiservices.c
```

content_c

In the **Debug** tab, you should see the API request. It may take a short time to become visible.

2. In **Debug**, click on the GET request, and then click **EV-QueryParamTest**.

In the **EV-QueryParamTest** details pane you should see that the variable **username** has been set to **joe**, and the variable **password** has been set to **abc123**.



If this had been a real password, you probably would not want it to be visible users of the Debug tool.

Task 4. Update the policy to use a private variable

In this task, you update the ExtractVariables policy to use a private variable so the value is not visible while tracing.

1. Return to the **Develop** tab and select **Policies > EV-QueryParamTest**.

2. Change the policy configuration to the following:

```
<ExtractVariables continueOnError="false" enabled="true"
name="EV-QueryParamTest">
  <QueryParam name="user">
    <Pattern ignoreCase="true">{username}</Pattern>
  </QueryParam>
  <QueryParam name="pw">
    <Pattern ignoreCase="true">{private.password}</Pattern>
  </QueryParam>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <Source clearPayload="false">request</Source>
</ExtractVariables>
```

Instead of **password**, the variable name for the pw query parameter is now **private.password**.
The **username** variable has remained unchanged.

This will change the resulting variable for the **pw** query parameter to **private.password**. No variable that starts with *"private."* will be shown in the debug tool.

3. Click **Save**, and then click **Save as New Revision**.

4. Click **Deploy**.

5. To specify that you want the new revision deployed to the eval environment, click **Deploy**.

6. Click **Confirm**.

   Wait for the deployment to complete.

Task 5. Verify that the private variable's value cannot be seen

In this task, you verify that the value of the variable private.password cannot be seen in the debug tool.

Start a debug session

1. Click the **Debug** tab, and then click **Start Debug Session**.

2. In the **Start debug session** pane, on the Environment dropdown, select **eval**.

3. Click **Start**.

Test the API proxy

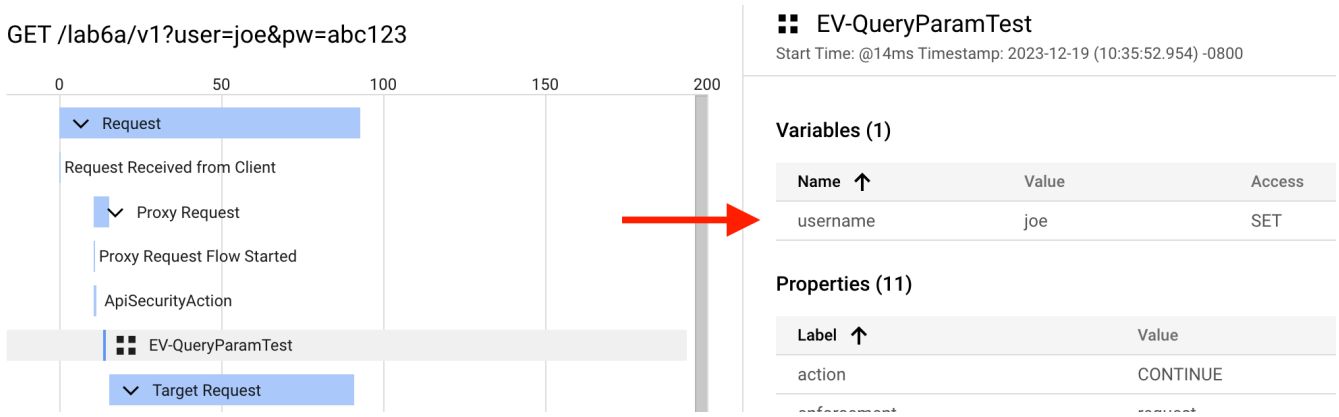
1. In Cloud Shell, send the following curl command:

```
curl -X GET "https://api-test-${GOOGLE_CLOUD_PROJECT}.apiservices.c
```

content_c

2. In **Debug**, click on the GET request, and then click **EV-QueryParamTest**.

   In the **EV-QueryParamTest** details pane you should see that the variable **username** has been set to **joe**, but the variable **password** is not visible.



> **Note:** If you copy the private variable into another non-private variable, that new variable would be visible in the debug tool.

Task 6. Create a debug mask configuration for a proxy

In this task, you create a debug mask to mask a variable in the debug tool.

This task will use the Apigee API to create a debug mask. There is a single debug mask associated with an environment.

1. Open a new browser tab in the same window and navigate to the Apigee API documentation page for updating a debug mask.

   This page shows how the updateDebugMask operation for an environment is called. You can use this page to make the API call to update the debug mask. You can also find the other Apigee API calls that are available.

   You will be using curl to make the API calls.

2. In Cloud Shell, make the following call:

   ```
   gcloud auth print-access-token
   ```

   content_c

   gcloud auth print-access-token is a gcloud command that prints the access token for the logged-in user. You logged in as the Qwiklabs user, so the token that is printed uses the permissions of the Qwiklabs user. Because the Qwiklabs user has the role of Owner on the project, the Qwiklabs user has organization admin permissions on the Apigee organization.

3. In Cloud Shell, make the following curl call to retrieve the debug mask for the eval environment:

   ```
   curl -H "Authorization: Bearer $(gcloud auth print-access-token)" -
   ```

   content_c

   The token will be provided to the Apigee API in an Authorization header. The response should look similar to this:

   ```
   {
       "name": "organizations/qwiklabs-gcp-03-ffaa428b506d/environments/eval/d
   }
   ```

   The debug mask contains the name of the debug mask plus any variables or XPath or JSONPath paths that have been configured to be masked. When the environment is created, there are no masked items.

4. To update the debug mask, use the following curl call:

```
curl -H "Authorization: Bearer $(gcloud auth print-access-
token)" -X PATCH
"https://apigee.googleapis.com/v1/organizations/${GOOGLE_CLOUD_PRO
-H "Content-Type: application/json" -d '{ "variables": [
"request.header.securitycode" ], "requestJSONPaths": [ "$.ccnum"
] }' | json_pp
```

content_c

The response to the curl call should look similar to this:

```
{
    "name": "organizations/qwiklabs-gcp-03-ffaa428b506d/environments/eval/d
    "requestJSONPaths": [
        "$.ccnum"
    ],
    "variables": [
        "request.header.securitycode"
    ]
}
```

This is the new debug mask. The **PATCH** command updated the debug mask with two items:

- A variable named **request.header.securitycode**
- A request JSONPath of **$.ccnum**

A header named **securitycode** would be masked. The JSONPath indicates that any incoming JSON request payload with the variable **ccnum** at the top level will be masked.

5. Return to the **Debug** tab and start a new debug session.

6. In Cloud Shell, send the following curl command:

```
curl -X PATCH "https://api-
test-${GOOGLE_CLOUD_PROJECT}.apiservices.dev/lab6a/v1?
user=joe&pw=abc123" -H "Content-Type: application/json" -H
```

content_c

```
"SecurityCode: secret" -d '{ "ccnum": "0123-4567-7654-3210" }' |
json_pp
```

Note: Note that header names are case-insensitive. Any capitalization of securitycode would be masked. Fields within JSON are case-sensitive, so the case must match exactly for a JSON field to be masked.

7. In **Debug**, click the **PATCH** request, and then click **Proxy Request Flow Started**.

   In the **Proxy Request Flow Started** pane, you can see that the **securitycode** header and the **ccnum** JSON fields have been replaced with asterisks.

8. To update the debug mask, use the following curl call:

```
curl -H "Authorization: Bearer $(gcloud auth print-access-
token)" -X PATCH
"https://apigee.googleapis.com/v1/organizations/${GOOGLE_CLOUD_PRO
replaceRepeatedFields=true" -H "Content-Type: application/json"
-d '{ "variables": [], "requestJSONPaths": [] }' | json_pp
```

   The response should look similar to:

```
{
    "name": "organizations/qwiklabs-gcp-03-ffaa428b506d/environments/eval/d
}
```

   The **replaceRepeatedFields** query parameter being equal to true indicates that the **variables** and **requestJSONPaths** arrays should replace what is in the debug mask instead of being appended to it. This will replace the masked fields with empty arrays, resetting the debug mask to empty.

9. Return to the **Debug** tab and start a new debug session.

10. Send this curl command again:

```
curl -X PATCH "https://api-
test-${GOOGLE_CLOUD_PROJECT}.apiservices.dev/lab6a/v1?
user=joe&pw=abc123" -H "Content-Type: application/json" -H
"SecurityCode: secret" -d '{ "ccnum": "0123-4567-7654-3210" }' |
json_pp
```

content_c

11. In **Debug**, click the **PATCH** request, and then click **Proxy Request Flow Started**.

    You can now see the **ccnum** JSON field and the **securitycode** header, because the debug masks for those are no longer in effect.

Congratulations!

In this lab, you used a private variable and a debug mask to hide data when debugging an API proxy.

End your lab