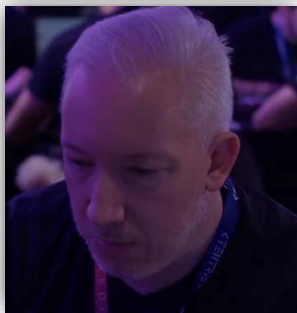


# **ESIEA Secure Edition 2023**

## **Google Apps Script**

---



Nicolas Kovacs

Team Leader Cloud Team

@lestutosdenico / Nicknam3



Sébastien Rolland

R&D Engineer Cloud Team

@Blindevy



- What is Google Apps Script?
- How to deploy a Google Apps Script?
- Authorization for Google Services
- Attack scenarios:
  - #1: Send documents by using forms
  - #2: Google Drive content from the victim to attacker's account
  - #3: Access Google Drive via permission change
  - #4: Send Google Drive documents to the attacker's mailbox
  - #5: Exfiltrate documents from a workstation to Google Sheet
  - #6: Implement persistent Google Workspace access
- Demo
- Conclusion

# What is Google Apps Script (GAS)



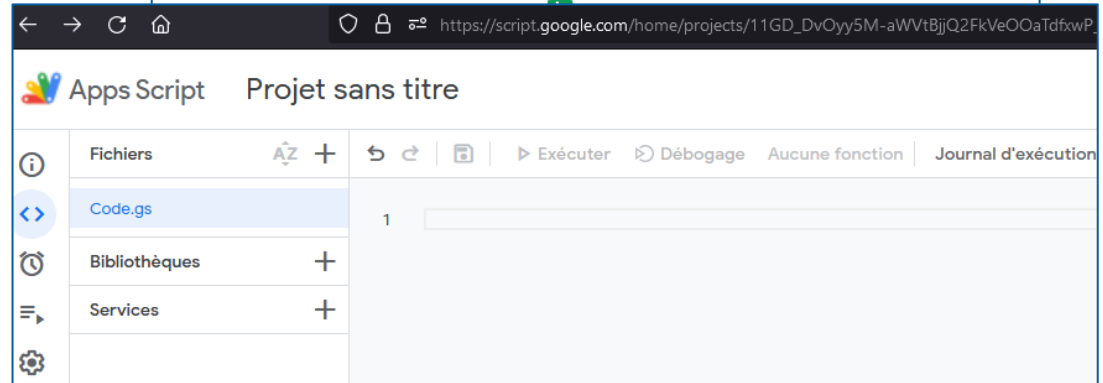
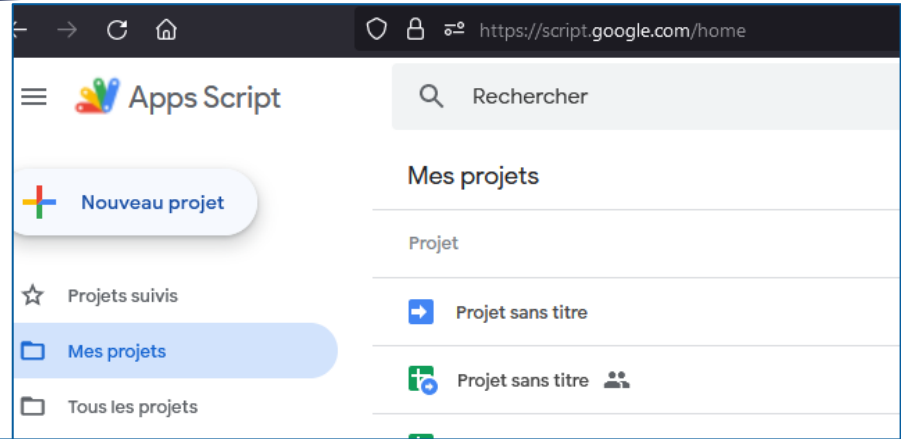
# What is Google Apps Script

- Application development platform
- Interact with Google applications: Gmail, Google Drive, Google Docs, Google Sheets, etc.) and Google Workspace
- Google Workspace:
  - Cloud-based suite that provides a variety of tools and applications
  - Help businesses and organizations communicate, collaborate, and get work done more efficiently
  - Includes applications such as Gmail, Google Drive, Google Docs, Google Sheets...
  - Designed for large organizations with more advanced security and management needs



# What is Google Apps Script

- To create a GAS project:
  - <https://script.google.com/home>
- GAS documentation:
  - <https://developers.google.com/apps-script/>
- GAS API:
  - <https://developers.google.com/apps-script/api/concepts>





# What is Google Apps Script

- Apps Script supports V8 JavaScript engine (Rhino is not used anymore)
- A lot of classes and methods are available
- Google Apps Script has built-in support for many Google APIs including Google Drive, Google Sheets, Google Calendar, etc.
- Import library feature

<code>base64Decode(encoded)</code>	<code>Byte[]</code>
<code>base64Decode(encoded, charset)</code>	<code>Byte[]</code>
<code>base64DecodeWebSafe(encoded)</code>	<code>Byte[]</code>
<code>base64DecodeWebSafe(encoded, charset)</code>	<code>Byte[]</code>
<code>base64Encode(data)</code>	<code>String</code>
<code>base64Encode(data)</code>	<code>String</code>
<code>base64Encode(data, charset)</code>	<code>String</code>
<code>base64EncodeWebSafe(data)</code>	<code>String</code>
<code>fetch(url)</code>	<code>HTTPResponse</code>
<code>base64EncodeWebSafe(d</code>	<code>fetch(url, params)</code> <code>HTTPResponse</code>
<code>base64EncodeWebSafe(d</code>	<code>fetchAll(requests)</code> <code>HTTPResponse[]</code>
	<code>getRequest(url)</code> <code>Object</code>
	<code>getRequest(url, params)</code> <code>Object</code>

# **How to deploy a Google Apps Script?**



# Hello() function



- Create a project on <https://script.google.com/>
- **gs** extension
- Execute **hello()** function:

The screenshot shows the Google Script Editor interface for a project named "ESE 2023". The top bar includes a "Déployer" button and a dropdown menu. The main editor area contains a JavaScript function:

```
1 function Hello() {  
2   Logger.log("Hello ESE");  
3 }  
4
```

Below the editor, the "Journal d'exécution" (Execution Log) is visible, showing three entries:

Journal d'exécution		
17:28:38	Avis	Exécution démarrée
17:28:38	Infos	Hello ESE
17:28:38	Avis	Exécution terminée

The first and third rows of the log are highlighted with a red border.

# *decodeBase64()* function



- Execute *decodeBase64()* function on <https://script.google.com/>

ESE 2023

Déployer

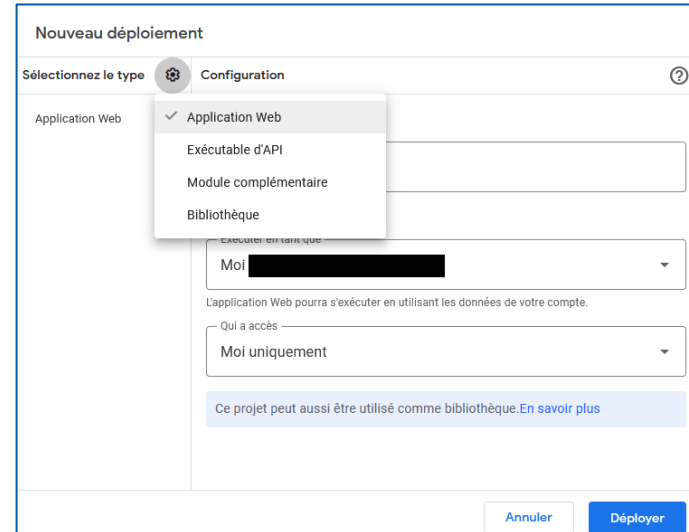
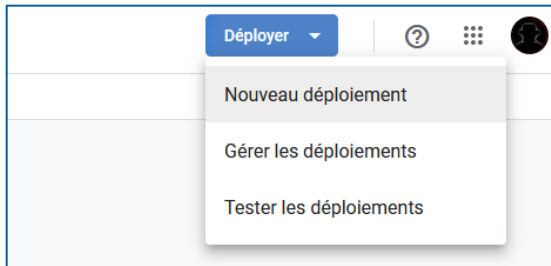
⌕ + ↶ ↷ 📄 ▶ Exécuter ⚙ Débogage decodeBase64 ▼ Journal d'exécution

```
1 function decodeBase64(encodedText){
2   var decodedText := Utilities.base64Decode(encodedText);
3   var decodedString := String.fromCharCode.apply(null, decodedText);
4   Logger.log("Decoded text:" + decodedString);
5 }
6
7 var encodedText := "SGVsbG8gRVNFICE=";
8 decodeBase64(encodedText);
```

Journal d'exécution

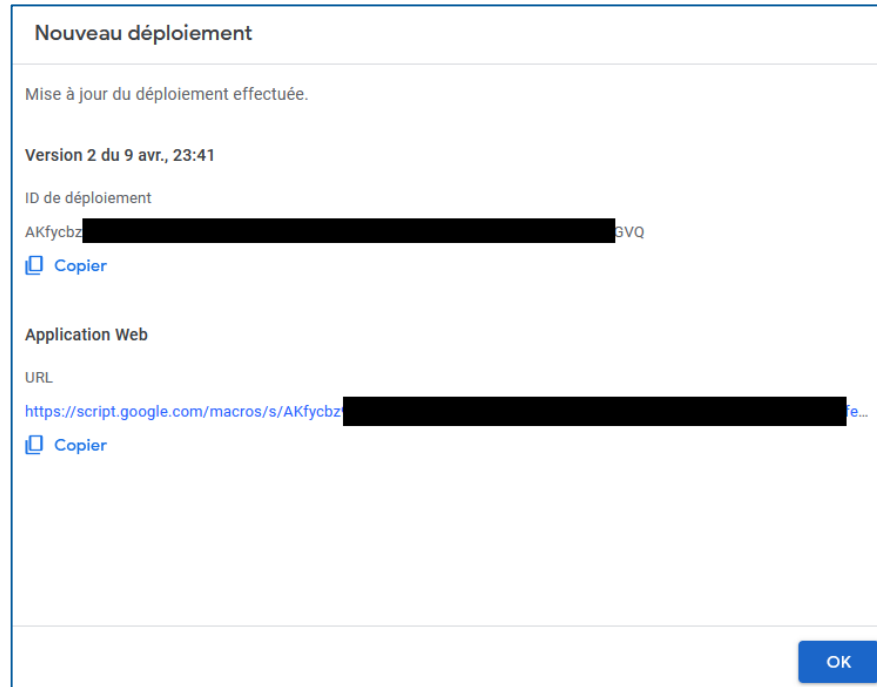
17:26:39	Avis	Exécution démarrée
17:26:39	Infos	Decoded text:Hello ESE !
17:26:39	Infos	Decoded text:

- Several configuration choices for deployment:
  - **Web application:** deploy a standalone web application
  - **API Executable:** deploy an Apps Script as an API
  - **Add-on:** deploy an Apps Script as a Google Workspace add-on (shared on Google Workspace Marketplace)
  - **Library:** share code across multiple Apps Script projects (distributed via a script ID)





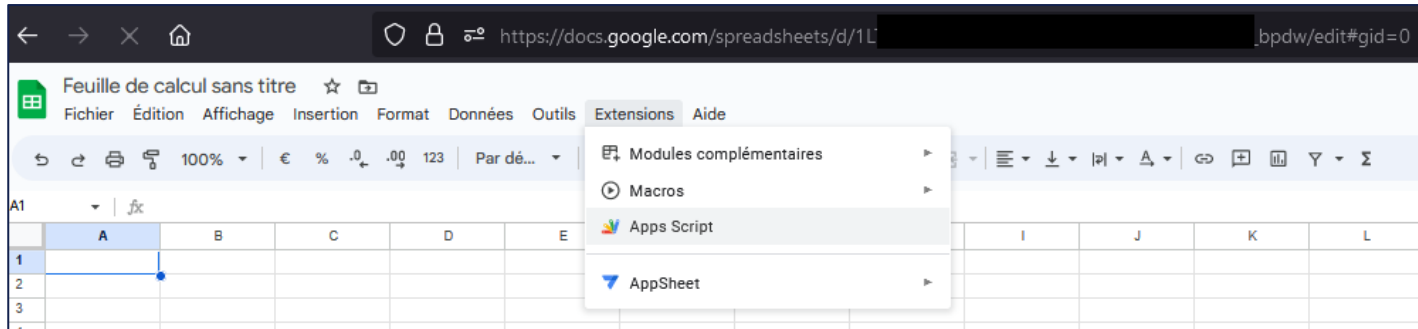
- A specific URL (with a unique ID) is generated by the Google Apps Script service:





# Deploy GAS via Google Sheets

- It is also possible to deploy an Apps Script directly on Google applications like Google Sheets:





# Simple script to fetch a URL

- Create a request on <https://www.quarkslab.com>
- Retrieve response
- Get the results on *A1* cell

The image shows a Google Apps Script editor window titled "Apps Script QITS 2023" with a file named "Code.gs". The script is as follows:

```
1 function requeteWeb() {  
2   var url = "https://www.quarkslab.com";  
3   var response = UrlFetchApp.fetch(url);  
4   var codeHtml = response.getContentText();  
5   var feuille = SpreadsheetApp.getActiveSpreadsheet().insertSheet("A1");  
6   var cellule = feuille.getRange("A1");  
7   var partieHtml = "";  
8   var longueurMaximale = 50000;  
9   for (var i = 0; i < codeHtml.length; i += longueurMaximale) {  
10    partieHtml = codeHtml.substr(i, longueurMaximale);  
11    cellule.setValue(partieHtml);  
12    cellule = cellule.offset(Math.floor(partieHtml.length / longueurMaximale), 0, 1, 1);  
13  }  
14  Logger.log("Récupération du code HTML terminée");  
15 }
```

The script is executed, and the results are shown in a Google Sheets window titled "Feuille de calcul sans titre". The content of the spreadsheet is the HTML of the Quarkslab website, starting with:

```
<!doctype html>  
<html lang="en-GB">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1">  
  <link rel="profile" href="https://gmpg.org/xfn/11">  
  <meta name="robots" content="index, follow, max-image-preview:large, max-snippet:-1, max-video-preview:-1">  
  <link rel="alternate" href="https://www.quarkslab.com/fr/accueil/" hreflang="fr" />  
  <link rel="alternate" href="https://www.quarkslab.com/" hreflang="en" />
```



# Useful functions for Web Applications

- ***doGet(e)***: called when an HTTP GET request is sent to the endpoint defined by the application
- ***doPost(e)***: called when an HTTP POST request is sent to the endpoint defined by the application
- The *e* argument represents an event parameter that can contain information about any request parameters:
  - ***e.queryString***: query string of the URL (*?user=qits&i=1337&i=31337*)
  - ***e.parameter***: an object of key/value pairs that correspond to the request parameters (*{“user”: “qits”, “i”: “1337”}*)
  - ***e.parameters***: similar to *e.parameter* but with an array of values for each key (*{“user”: [“qits”], “i”: [“1337”, “31337”]}*)
  - ***e.pathInfo***: URL path after */exec*
  - ***e.contentLength***: length of the request body for POST requests
  - ***e.postData.contents***: content text of the POST body
  - etc.



# Triggers

- Triggers are functions that are automatically executed in response to specific events or triggers
- Triggers are useful for automating tasks and simplifying repetitive processes
- Some common triggers include time triggers, form triggers, change triggers, and email triggers:
  - Time triggers: define functions to run at specific times or regular intervals
  - Form triggers: execute a function in response to a Google Sheets, Google Forms, or Google Slides form submission
  - Change triggers: automatically runs whenever a specific cell is modified in a Google Sheets spreadsheet
  - Email triggers: automatically send emails in response to specific events





# Limitations

- Some limitations:

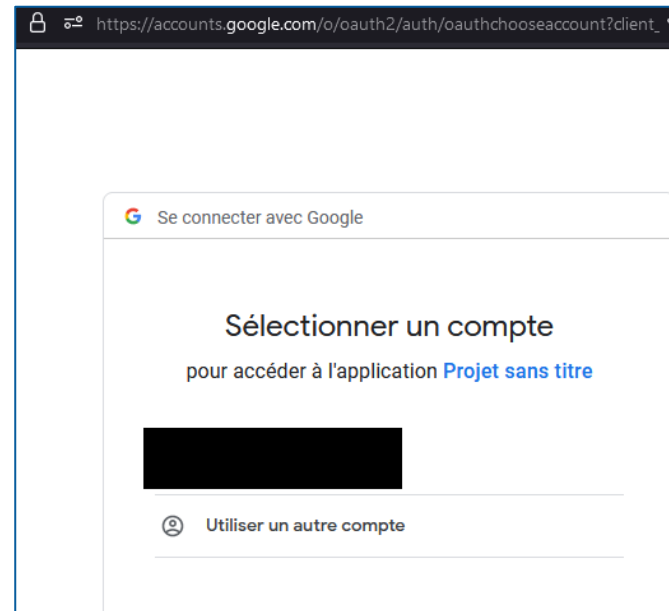
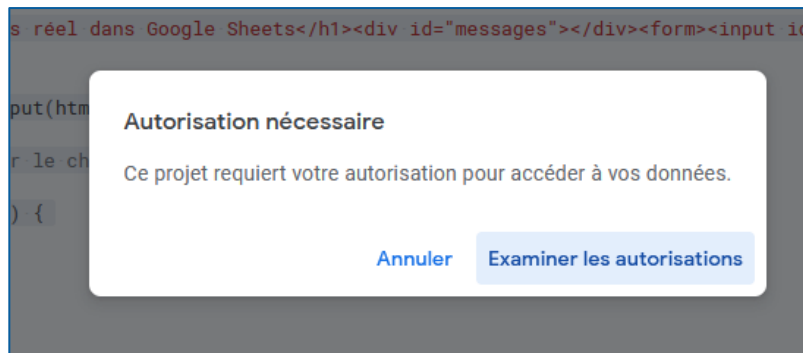
Feature	Consumer (e.g., gmail.com) and G Suite free edition (legacy)	Google Workspace accounts
Calendar events created	5,000 / day	10,000 / day
Contacts created	1,000 / day	2,000 / day
Documents created	250 / day	1,500 / day
Files converted	2,000 / day	4,000 / day
Email recipients per day	100* / day	1,500* / day
Email recipients per day within domain	100* / day	2,000 / day
Email read/write (excluding send)	20,000 / day	50,000 / day
Groups read	2,000 / day	10,000 / day
JDBC connection	10,000 / day	50,000 / day
JDBC failed connection	100 / day	500 / day
Presentations created	250 / day	1,500 / day
Properties read/write	50,000 / day	500,000 / day
Slides created	250 / day	1,500 / day
Spreadsheets created	250 / day	3,200 / day
Triggers total runtime	90 min / day	6 hr / day
URL Fetch calls	20,000 / day	100,000 / day

Feature	Consumer (e.g., gmail.com) and G Suite free edition (legacy)	Google Workspace accounts
Script runtime	6 min / execution	6 min / execution
Custom function runtime	30 sec / execution	30 sec / execution
Simultaneous executions	30 / user	30 / user
Email attachments	250 / msg	250 / msg
Email body size	200 KB / msg	400 KB / msg
Email recipients per message	50 / msg	50 / msg
Email total attachments size	25 MB / msg	25 MB / msg
Properties value size	9 KB / val	9 KB / val
Properties total storage	500 KB / property store	500 KB / property store
Triggers	20 / user / script	20 / user / script
URL Fetch response size	50 MB / call	50 MB / call
URL Fetch headers	100 / call	100 / call
URL Fetch header size	8 KB / call	8 KB / call
URL Fetch POST size	50 MB / call	50 MB / call
URL Fetch URL length	2 KB / call	2 KB / call

# Authorization for Google Services

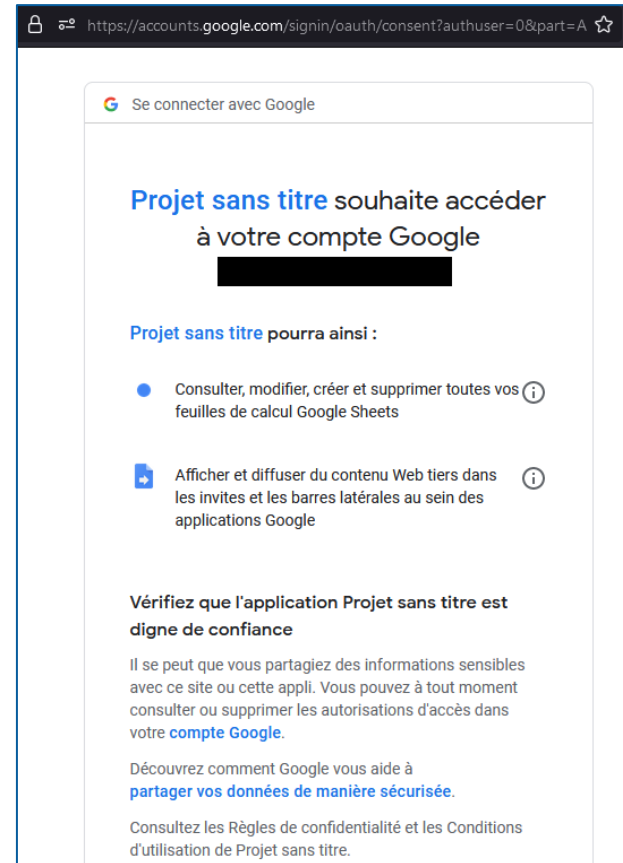
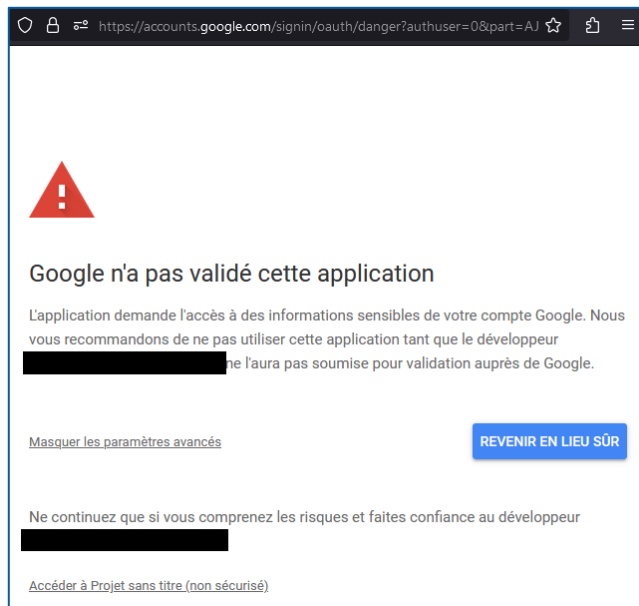
# Authorization 1/2

- Apps Script determines the authorization scopes automatically
- The analysis is based on a scan of the code
- If a script needs authorization, you'll see one of the authorization dialogs shown here when it is run



# Authorization 2/2













- When the application is accessed by a user, Google displays two alerts:



# Exception



- But Google makes an exception when the application is created in an enterprise tenant and accessed by an enterprise account

	Client validé	L'éditeur est un compte Google Workspace du client A.	Le script se trouve dans un Drive partagé du client A	L'éditeur est un compte Gmail
L'utilisateur est un compte Google Workspace du client A.	Flux d'authentification normal 	Flux d'authentification normal 	Flux d'authentification normal 	Flux d'authentification validé 
L'utilisateur est un compte Google Workspace <u>non</u> du client A.	Flux d'authentification normal 	Flux d'authentification non validé 	Flux d'authentification non validé 	Flux d'authentification validé 
L'utilisateur est un compte Gmail <sup>1</sup>	Flux d'authentification normal 	Flux d'authentification non validé 	Flux d'authentification non validé 	Flux d'authentification validé 



- **Permission configuration:**
  - **Execute the app as me:** In this case, the script always executes as the owner of the script, no matter who accesses the web app
  - **Execute the app as user accessing the web app:** In this case, the script runs under the identity of the active user using the web app

### Nouveau déploiement

Sélectionnez le type

Configuration

Application Web

Description

Nouvelle description

Application Web

Exécuter en tant que

Utilisateur accédant à l'application Web

Moi

Utilisateur accédant à l'application Web

Ce projet peut aussi être utilisé comme bibliothèque.[En savoir plus](#)

Annuler

Déployer



# Application access

- Who can access the application:
  - Only me:** the application creator
  - Anyone with Google account:** any user authenticated with Google

### Nouveau déploiement

Sélectionnez le type

Configuration

Application Web

Description

Nouvelle description

Application Web

Exécuter en tant que

Utilisateur accédant à l'application Web

L'application Web demandera aux utilisateurs l'autorisation de s'exécuter en utilisant les données de leur compte.

Qui a accès

Moi uniquement

Moi uniquement

Tout utilisateur possédant un compte Google

Annuler

Déployer

# Attack scenarios



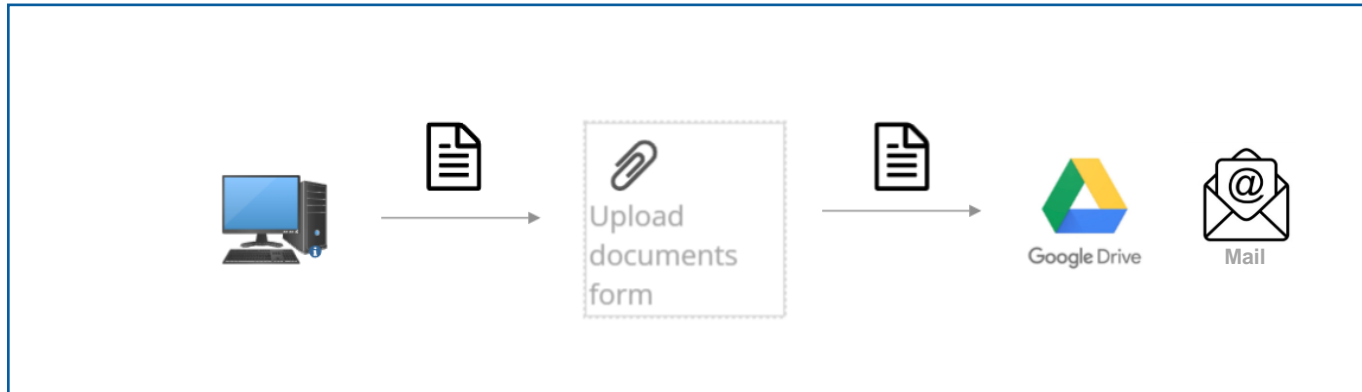


# The customer's needs

1. Is it possible to exfiltrate data by using Google Apps Script?
2. Even if Cloud Access Security Broker (CASB) is used?


# Scenario #1

- **Intentional** exfiltration by internal user:
  - Create simple form to upload document
  - The uploaded documents are sent to the attacker's Google Drive and/or by email





# Scenario #1



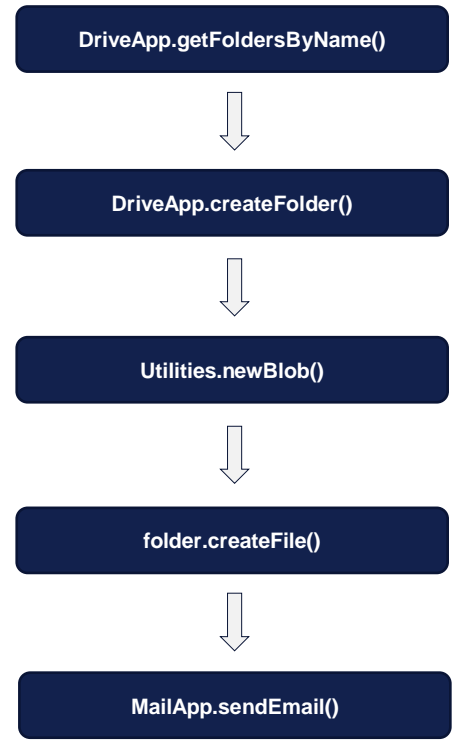
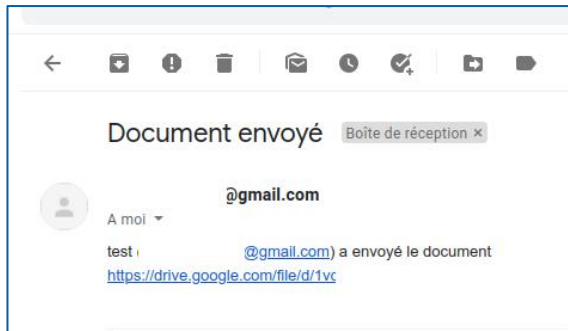
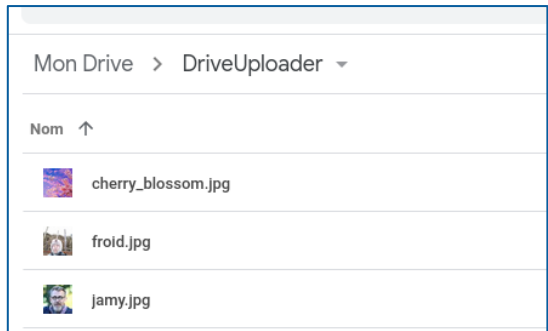
Nom

Mail

Parcourir...

Aucun fichier sélectionné.

Envoyer le document



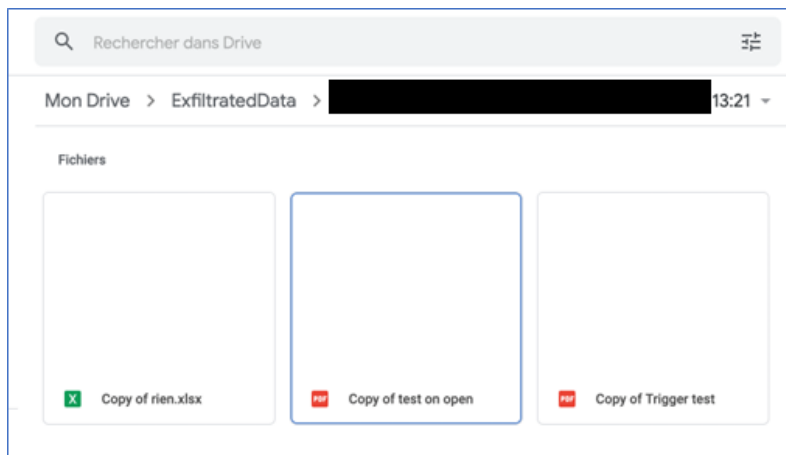
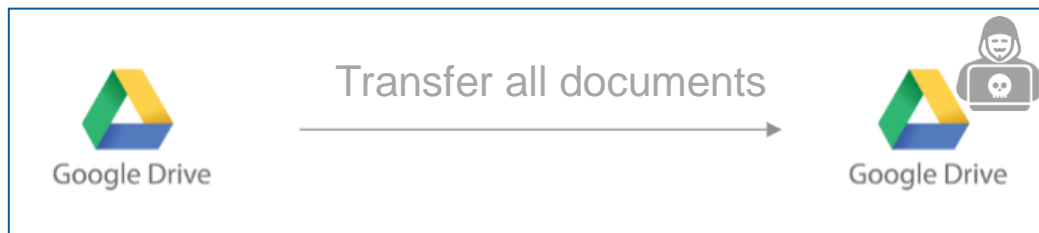


## Scenario #2

- The victim accesses a malicious GAS application
- The executed script lists all files in the victim's Google Drive
- These files are then transformed into blobs, byte array, compressed and encoded in base64
- The base64-encoded string is divided into 50 000 characters chunks (limit for one cell on Google Sheet) and written into the cells of the column dedicated to the current file
- A second application rebuild the data exfiltrated in base64 and write the files to the Google Drive space of the attacker



# Scenario #2



## exfiltrate.gs:

```
Utilities.base64Encode(Utilities.gzip(  
  file.getBlob()).getBytes())
```



```
file.getBlob().getContentType()
```

## recover.gs:

```
Utilities.base64Decode(b64)
```



```
Utilities.newBlob()
```

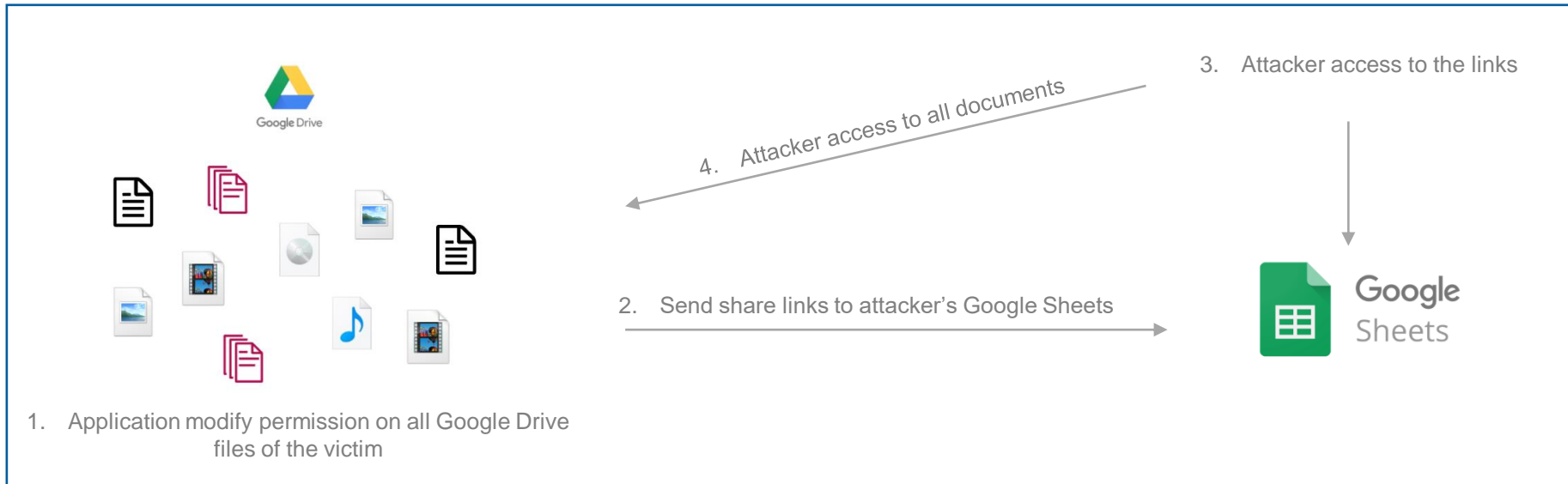


```
Utilities.ungzip()
```



# Scenario #3

- The victim accesses a malicious GAS application
- The permissions of all the victim's Google Drive files are changed
- The links of each documents are sent to a Google Sheets belonging to the attacker



# Scenario #3



Feuille de calcul sans titre										
	A	B	C	D	E	F	G	H	I	J
1	Feuille de calcul	06/07/2022 15:5	1024	<a href="https://docs.google.com/spreadsheets/d/1Sxhik9w...">https://docs.google.com/spreadsheets/d/1Sxhik9w...</a>						<a href="#">drivesdk</a>
2	Feuille de calcul	04/07/2022 14:3	10265	<a href="https://docs.google.com/spreadsheets/d/1Sxhik9w...">https://docs.google.com/spreadsheets/d/1Sxhik9w...</a>						<a href="#">resdk</a>
3	Copie de Feuille	04/07/2022 21:2	1024	<a href="https://docs.google.com/spreadsheets/d/1Sxhik9w...">https://docs.google.com/spreadsheets/d/1Sxhik9w...</a>						<a href="#">vesdk</a>
4	poc5	04/07/2022 20:4	1024	<a href="https://docs.google.com/spreadsheets/d/1Sxhik9w...">https://docs.google.com/spreadsheets/d/1Sxhik9w...</a>						<a href="#">ivesdk</a>
5	POC	04/07/2022 10:3	1024	<a href="https://docs.google.com/spreadsheets/d/1Sxhik9w...">https://docs.google.com/spreadsheets/d/1Sxhik9w...</a>						<a href="#">rivesdk</a>
6	newFiles.zip	04/07/2022 10:5	13150388	<a href="https://drive.google.com/file/d/19nyQDa...">https://drive.google.com/file/d/19nyQDa...</a>						
7		28/06/2022 16:5	1024	<a href="https://docs.google.com/spreadsheets/d/1Sxhik9w...">https://docs.google.com/spreadsheets/d/1Sxhik9w...</a>						<a href="#">=drivesdk</a>
8		29/06/2022 16:0	1024	<a href="https://docs.google.com/spreadsheets/d/1Sxhik9w...">https://docs.google.com/spreadsheets/d/1Sxhik9w...</a>						<a href="#">?usp=drivesdk</a>
9		25/04/2022 15:4	1024	<a href="https://docs.google.com/spreadsheets/d/1Sxhik9w...">https://docs.google.com/spreadsheets/d/1Sxhik9w...</a>						<a href="#">sp=drivesdk</a>
10		21/06/2022 16:2	1024	<a href="https://docs.google.com/spreadsheets/d/1Sxhik9w...">https://docs.google.com/spreadsheets/d/1Sxhik9w...</a>						<a href="#">esdk</a>
11		29/06/2022 11:1	1024	<a href="https://docs.google.com/spreadsheets/d/1Sxhik9w...">https://docs.google.com/spreadsheets/d/1Sxhik9w...</a>						<a href="#">livesdk</a>
12		30/06/2022 18:2	142679	<a href="https://drive.google.com/file/d/1Sxhik9w...">https://drive.google.com/file/d/1Sxhik9w...</a>						
13		25/04/2022 15:4	1024	<a href="https://docs.google.com/spreadsheets/d/1Sxhik9w...">https://docs.google.com/spreadsheets/d/1Sxhik9w...</a>						<a href="#">=drivesdk</a>
14										
15										

DriveApp.GetFiles()



file.setSharing(DriveApp.Access.ANY  
ONE, DriveApp.Permission.EDIT)



**Vous avez partagé un élément**



rien.xlsx



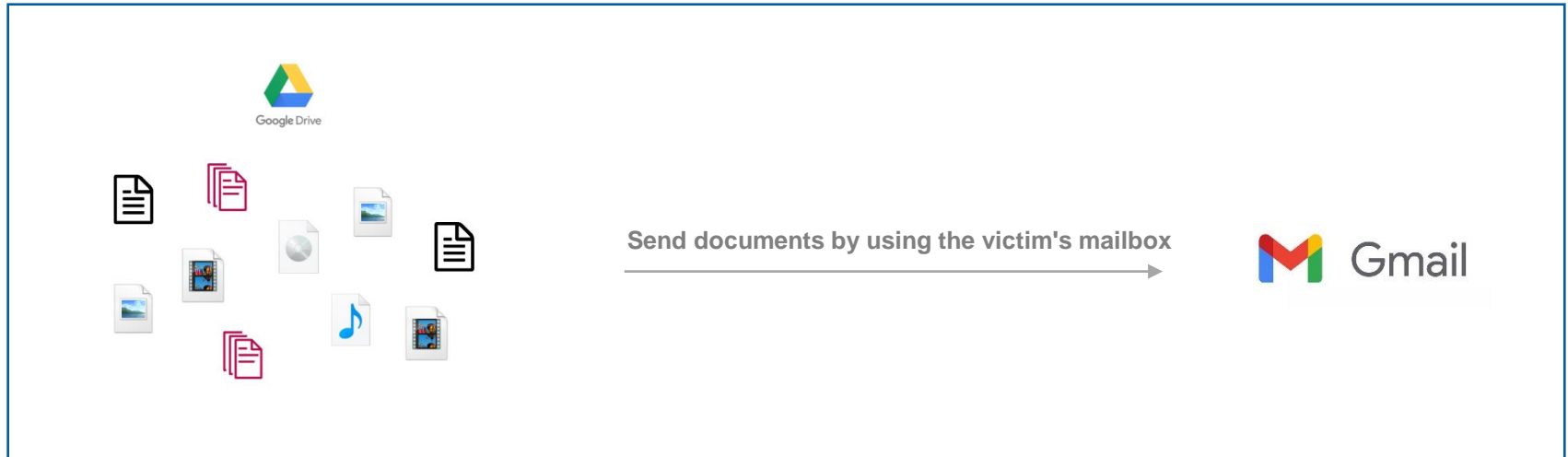
**Tous les  
internautes**

Modification autorisée



## Scenario #4

- The victim accesses a malicious GAS application
- All the documents in the Google Drive space are exfiltrated to the attacker's mailbox
- It is possible to compress documents in zip via Google Apps Script in order to send the documents in a single operation

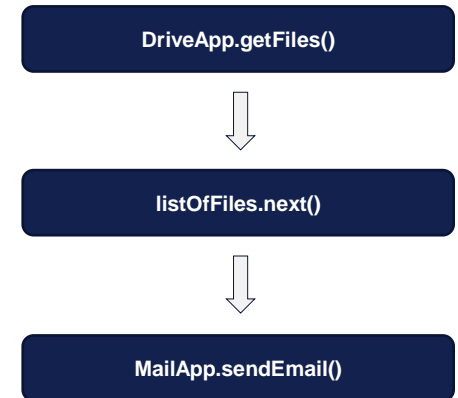






## Scenario #4

- Limitation:
  - 50MB attachment size limitation is applied by GAS



# Scenario #5

- This scenario differs from the previous ones because it is initiated from the victim's machine
- *Exfiltration.ps1* to list files, encode, create and send the chunks to a Google Sheets
- *Reveal.ps1* to reconstruct the files on the attacker's side



1. List all files or with specific extensions
2. Base64 encode + XOR
3. Chunk of 50,000 characters
4. Send the chunks to Google Sheets (with filenames + extensions)



Google  
Sheets

1. Google Sheets access
2. Retrieve all chunks for each files
3. XOR + Base64 decode
4. Files recovery







## Scenario #6

- Use triggers based on schedule, time or from the attached file event like opening or modification
- Triggers are linked to a Google App Scripts, this means victim needs write permissions
- Example scenario:
  - Checks every minute if the victim has received a mail with subject « *Security Alert* » from Google
  - If the condition is true, immediately delete the mail
- All of our scenarios can be provided with a trigger, like time trigger to keep persistence



# Scenario #6

Apps Script Trigger test Déployer ? ⋮ S

**Déclencheurs** 1 déclencheur déclencheurs affichés

+ Ajouter un filtre

Propriétaire	Dernière exécution	Déploiement	Événement	Fonction	Taux d'erreur
Moi	-	Head	Basé sur l'heure	runScript	-

`DocumentApp.getActiveDocument()`



`ScriptApp.getUserTriggers()`



`ScriptApp.newTrigger()`

# Demo

1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
30		
31		
32		
33		
34		
35		
36		
37		
38		
39		
40		
41		
42		
43		
44		

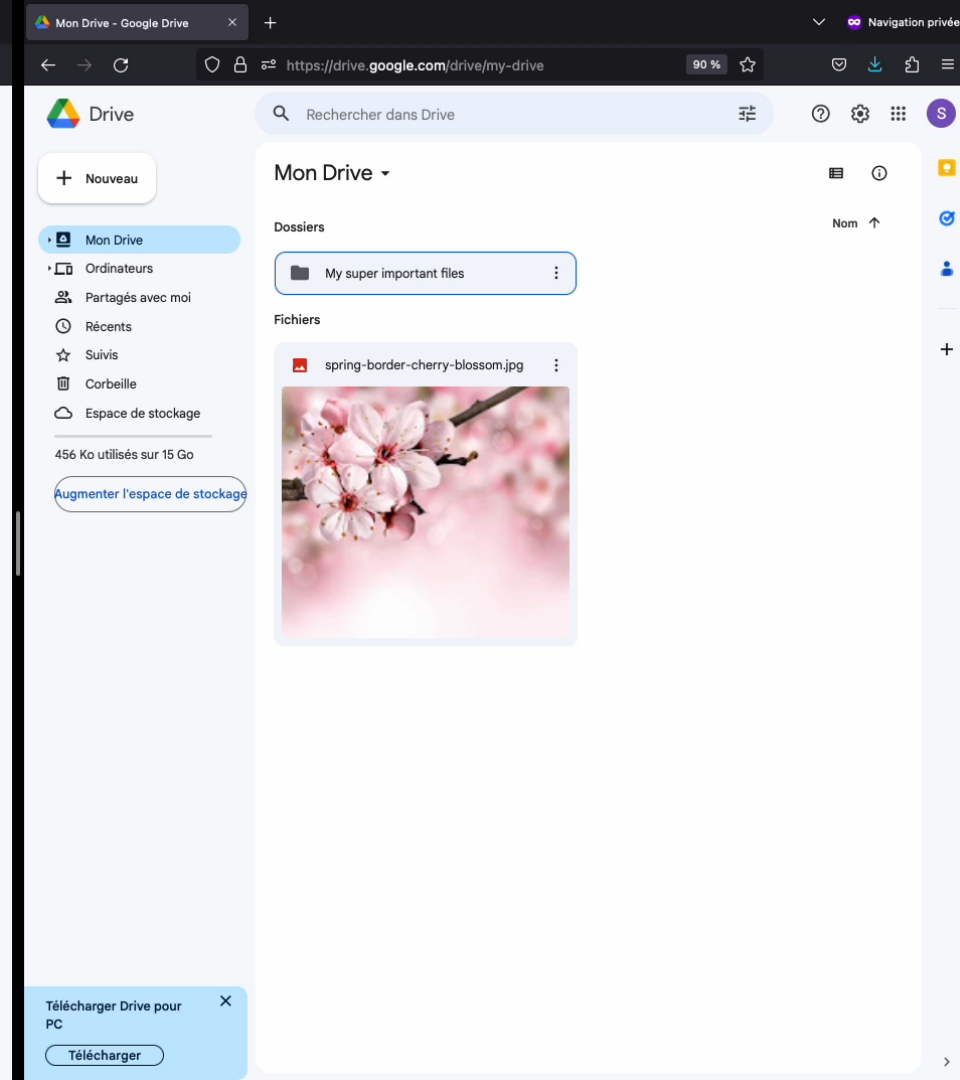
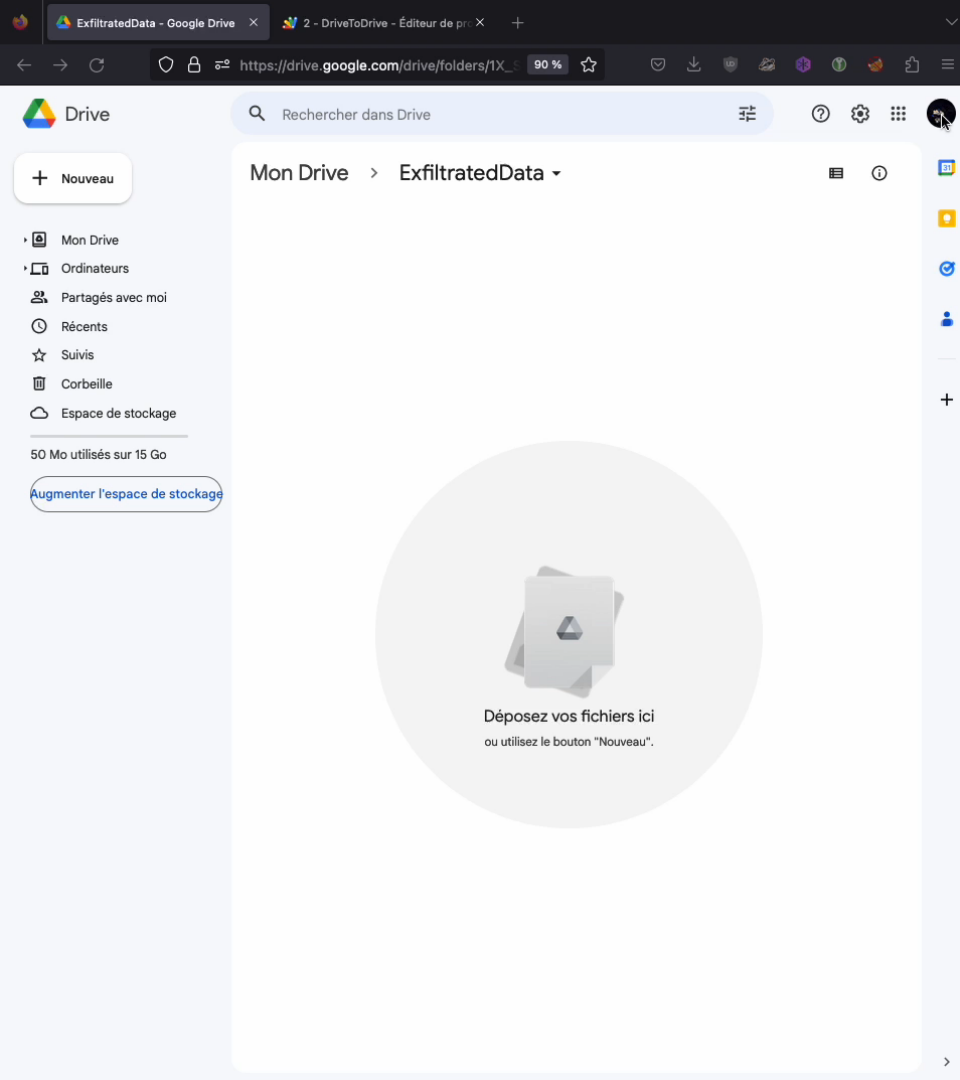
## Windows PowerShell

```
=====
===== Gdrive Exfiltrate =====
=====

1. To send all files (pdf, docx, doc, xlsx, xls) for the chosen drive to a Google Sheet.
2. To send files with the chosen extension (.extension format) and for the chosen drive to a Google Sheet.
3. To send the file of your choice to Google Sheet (format - C:\folder\file.ext).
Q: To exit

Choose an option: _
```

	Type	Taille
5:38	Script Windows P...	10 Ko
5:36	Script Windows P...	4 Ko





# Conclusion



# Conclusion

- Google Apps Script is a powerful tool to automate many tasks in Google applications
- It is possible to use the service to conduct a variety of attacks
- A lot of Google services are used by internal users and therefore potentially more difficult to detect by the blue team
- It is possible to restrict the use of Google Apps Script for accounts using Google Workspace
- However, some protection services potentially misconfigured like Cloud Access Security Broker (CASB) are blind to these attacks

# Bonus



# Basic reflected XSS scan via GAS

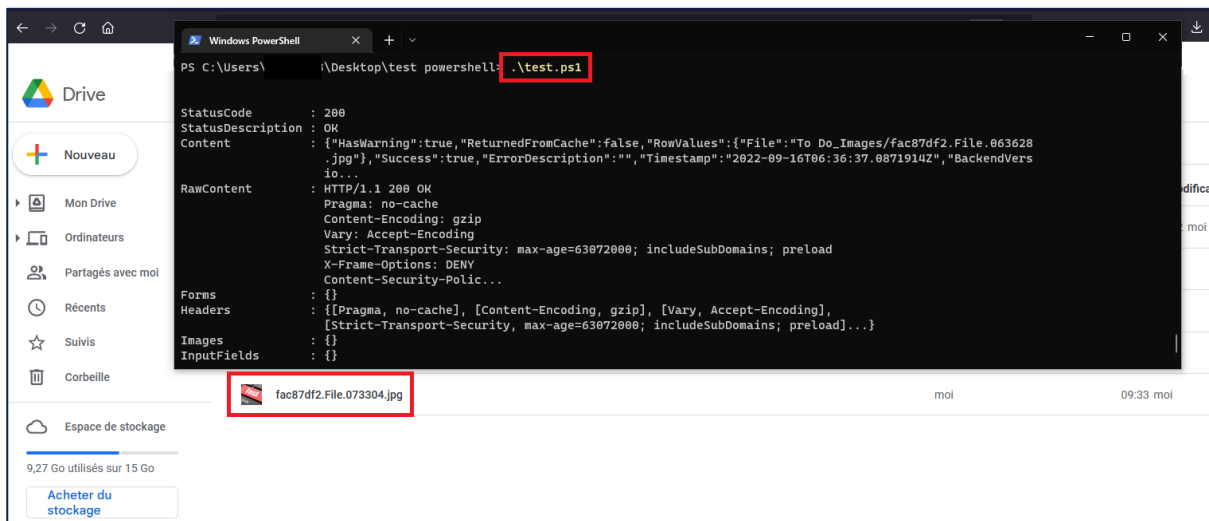
## Code snippet (Code.gs):

```
function doGet() {  
  var template = HtmlService.createTemplateFromFile('Page');  
  template.action = 'start';  
  return template.evaluate();  
}  
  
function processUrls(form) {  
  var urls = form.urls.split('\n');  
  var validUrls = [];  
  
  for (var i = 0; i < urls.length; i++) {  
    var url = urls[i].trim();  
  
    if (url === '') {  
      continue;  
    }  
  
    var scanUrl = url + '?q=icicoucou%3Cscript';  
    var response;  
  
    try {  
      response = UrlFetchApp.fetch(scanUrl, {  
        muteHttpExceptions: true,  
        validateHttpsCertificates: false  
      });  
    } catch (error) {  
      continue;  
    }  
  
    if (response.getResponseCode() !== 200) {  
      continue;  
    }  
  
    var content = response.getContentText();  
    if (!content.includes('icicoucou<script')) {  
      continue;  
    }  
  
    validUrls.push(url);  
    SpreadsheetApp.getActiveSpreadsheet().getSheets()[0].appendRow([url]);  
  }  
  
  var template = HtmlService.createTemplateFromFile('Page');  
  template.validUrls = validUrls;  
  template.action = 'complete';  
  return template.evaluate();  
}
```

## Code snippet (Page.html):

```
<!DOCTYPE html>  
<html>  
  <head>  
    <base target="_top">  
    <script>  
      function processForm() {  
        var form = document.getElementById('urls-form');  
        var data = new FormData(form);  
  
        google.script.run.withSuccessHandler(displayResults).processUrls(  
          Object.fromEntries(data.entries());  
        );  
  
        function displayResults(result) {  
          if (result.action === 'complete') {  
            document.getElementById('message').textContent = 'Done';  
          }  
        }  
      }  
    </script>  
  </head>  
  <body>  
    <form id="urls-form" onsubmit="event.preventDefault(); processForm();">  
      <label for="urls">URLs:</label>  
      <textarea id="urls" name="urls" rows="10" cols="50"></textarea>  
      <br>  
      <button type="submit">Submit</button>  
    </form>  
    <div id="message"></div>  
  </body>  
</html>
```

- AppSheet is a no-code platform that permit to anyone to build mobile application without writing a line of code
- Some scenarios can used on this service like exfiltration:





# Admin SDK Directory Service

- It is also possible to manage devices, groups, users and other entities in Google Workspace domains (Admin SDK Directory Service)
  - <https://developers.google.com/admin-sdk/directory/reference/rest>

## List all the groups in the domain

```
function listAllGroups() {
  let pageToken;
  let page;
  do {
    page = AdminDirectory.Groups.list({
      domain: 'example.com',
      maxResults: 100,
      pageToken: pageToken
    });
    const groups = page.groups;
    if (!groups) {
      console.log("No groups found.");
      return;
    }
    // Print group name and email.
    for (const group of groups) {
      console.log("%s (%s)", group.name, group.email);
    }
    pageToken = page.nextPageToken;
  } while (pageToken);
}
```

## Add user to domain admin

```
function addAdmin(email) {
  var user = {
    primaryEmail: email,
    isAdmin: true
  };
  AdminDirectory.Users.update(user, email,
    {makeAdmin: true});
}
```

**Thanks**  
**Any question?**