Spirion Extensions

Spirion Automation: Google Drive Labels & Share Permissions

Table of Contents

Spirion Automation: Google Drive Labels & Share Permissions

Introduction

Centralizing sensitive data discovery with Spirion ensures consistent, highly accurate outcomes across systems ranging from end-user workstations to cloud-hosted SaaS solutions. In doing so, information repositories can be managed cohesively without having to redundantly calibrate scanning tools built into individual walled gardens in hopes of arriving at identical results.

This extension described how Spirion's dynamic scripting capabilities can incorporate platform-specific extensibility through API (or CLI) interactions, giving data security administrators the best of both worlds – uniform visibility into their entire sensitive data footprint with the option to employ additional control mechanisms outside of Spirion itself.

Google Drive labels classify files hosted in Google Workspace. While they can be set manually by users or automatically via Google Workspace DLP rules, a programmatic approach through the Drive API offers seamless orchestration with broader security and privacy policies. Similarly, the same API can audit additional metadata characteristics, including permissions shared to internal or external accounts.

Common support for Python across APIs, SDKs, and Spirion scripts make it ideal for prototyping how to integrate match results from a Spirion discovery scan with meaningful functionality from third-party applications. The following information serves as reference for scripting automated API interactions with Google Drive.

Released July 2023, © 2023 Spirion LLC.

Requirements

Before working on the steps outlined in this document, please confirm the following:

- The Spirion console (SDM or SDP) is up-to-date and accessible.
- A Spirion endpoint agent hosted on a Windows 10 system is online, polling to its console.
- Python is installed on the Windows host running the Spirion agent.
    - **NOTE:** LocalSystem must be able to execute the Python script called by Spirion.

- Admin access to a Google Workspace environment with Drive labels is available.
    - **NOTE:** Drive labels are supported by the following editions: *Business Standard and Business Plus; Enterprise; Education Standard and Education Plus; Essentials, Enterprise Essentials, and Enterprise Essentials Plus; G Suite Business*.

Important Notes

Software Versions

SDP console version 22.Q3.1.226.1, SDM console version 11.8.2, Spirion agent version 12.5, and Python 3.11.1 were used to perform the steps described in this document.

Google Drive API

The configuration explained below references the Python implementation of Google's API to create a script that applies a Drive badged label to matches from Spirion search results.

Authentication is handled using a JSON Web Token (JWT) associated with a service account created in the GCP Console.

Process

Spirion can automate the execution of shell scripts with variables derived from scan results discovered by Spirion agents. In SDM, this is configured from the Action tab of any given Workflow, and from the Playbook editor in SDP.

To interact with Google Drive, the Spirion console is configured with a batch script that invokes a Python script hosted and executed locally by the Spirion agent(s) tasked with searching cloud storage. The Python script takes the Spirion match location (a Google Drive file path) as a command-line argument handled by the scripting engine's "%Location%" variable. A sample Python script is available on the Spirion website in file google_drive.txt.

Configuring Google Drive

Please ensure the following steps have been completed within the Google Cloud Platform (GCP) tenant prior to running the example script.

Creating a Service Account

From the GCP console…

1. Create (or select an existing) Google Cloud project.
2. Enable the Google Drive API for that project.
3. Browse to ***APIs & Services › Credentials*** and click on ***+ Create Credentials*** to create a service account.
4. Navigate to ***Keys*** for the service account, select ***Add Key*** and create a JSON private key.
    a. The .JSON file must be accessible to the system running the Spirion agent.
    b. This key is used by the example script, referencing in `C:\temp` by default.

Setting Labels & Permissions

From the Google Workspace admin console…

1. Manage Domain Wide Delegation for the service account created in the previous section.
    a. Add the scope specified in the Python script.
        i. E.g. https://www.googleapis.com/auth/drive.metadata
2. Select a Google Drive Label.
    a. This script uses a badged label.
3. Get the id values for the label as well as its field and choice(s).
    a. Refer to the Google Drive Labels API for guidance obtaining these values.
    b. Example values are plugged into the script's customization section for reference.

Scripting the Drive API

Using Google Drive's Python SDK, the example script attached to this document executes the following logic:

1. Get the Spirion match result location – a file path terminating to a Drive user account.
    a. E.g. "example.address@email.com/drive folder1/drive folder2/file.ext"

2. Extract the following from this location:
    a. Drive account email
    b. Breadcrumbs of the folder structure
        i. This is used for match validation in step 4.
    c. File name
3. Search Google Drive account (a) for the file (c).
4. Validate that the file returned via API is the same file referenced in the Spirion match.
5. Set the Drive label of the file confirmed in the previous step.
6. Get and log active file permissions associated with the match result.

NOTE: The Python sample file includes 5 variables that must be edited to reflect valid settings for the environment being used – `CREDS`, `PATH`, `LABEL_ID`, `FIELD_ID`, and `CHOICE_ID` – defined in the "Customization" section (lines 12 through 25).

Configuring the Spirion Agent

In addition to moving the .JSON key file over to the Spirion agent, also install the Python script's module dependencies for the Google Drive SDK:

```
pip install --upgrade google-api-python-client google-auth-httplib2 google-auth-oauthlib
```

NOTE: LocalSystem must have access to these modules.

Configuring Sensitive Data Platform (SDP)

If using the SDP console, perform the following:

Uploading a Script

1. From the main menu, select ***Settings***.
2. Navigate to ***Script Repository***.
3. Click the ***Actions*** button toward the top-right of SDP and select "Add Script."
4. Specify a ***name*** and ***description*** for the script.
5. Click the icon next to ***Upload Script*** and upload the following command as a batch file:

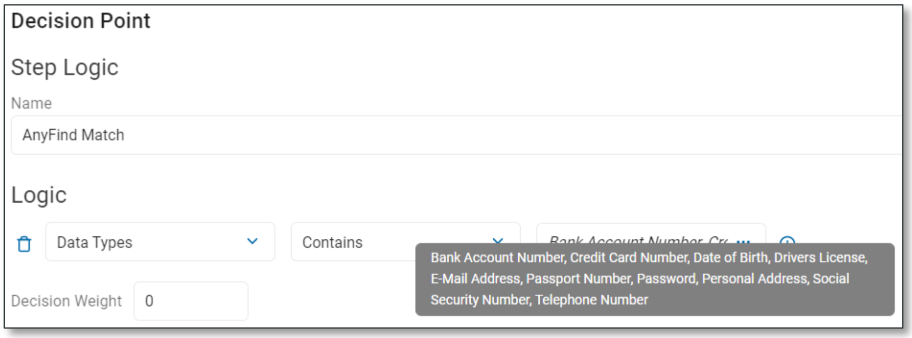
```
python c:\temp\google_drive.py %Location%
```

6. ***Save*** the uploaded script.

Creating a Playbook

The steps described below cover the relevant steps for automated script execution. For additional guidance, please refer to the Spirion KB Article: How to Write a Playbook.

1. From the main menu, select ***Scans***.
2. Navigate to ***Scan Playbooks***.
3. Click ***Add Playbook*** using the button from the top-right.
4. Enter a ***Name*** and ***Description*** as desired.
5. Under the ***Logic*** section of a ***Decision Point***, select the criteria that should trigger the rule.
    a. For example, specifying "Data Types" and selecting any (or all) of the built-in AnyFind search parameters would trigger this script for every location (e.g. file/document) with 1 or more of the selected data types.



Decision Point

Step Logic

Name: AnyFind Match

Logic

Data Types | Contains | Bank Account Number, Cr...

Bank Account Number, Credit Card Number, Date of Birth, Drivers License, E-Mail Address, Passport Number, Password, Personal Address, Social Security Number, Telephone Number

Decision Weight: 0

6. From the ***Select Action*** pulldown menu, select "Execute Script" for the action card that corresponds to the intended decision path (yes/no) per the logic specified in the previous step.
7. Under ***Select Script***, choose the script uploaded in the previous section.
8. Define additional Playbook logic as necessary before clicking the ***Save*** icon to finalize changes.

Searching Google Drive

Once the Playbook has been created, it needs to be associated with a scan policy targeted at Google Drive. Please reference the documentation under each step for additional guidance.

1. Add Google Drive as a ***Target*** after navigating to ***Data Assets and Targets*** from the ***Data Asset Inventory***.
    a. Spirion KB Article: Create a Google Drive Target
2. Create a new ***Scan*** by clicking the ***Add Scan*** button from the ***Scans*** screen.

3. Select the Playbook and Target configured per the guidance above.
    a. Spirion KB Article: Create a Scan to Search Google Drive

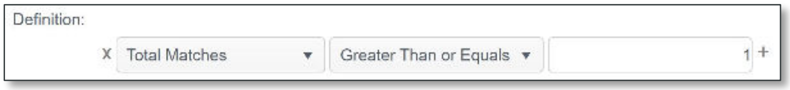
Configuring Sensitive Data Manager (SDM)

If using the SDM console, perform the following:

Creating a Workflow

The steps described below cover the relevant steps for automated script execution. For additional guidance, please refer to the Spirion KB Article: How to Write a Workflow Rule.

1. From the ***Workflows*** tab, select “Add” from the ***Rule*** pulldown menu.
2. Enter a ***Name*** and ***Description*** as desired.
3. On the ***Definition*** tab, select the criteria that should trigger the rule.
    a. For example, specifying “Total Matches Greater Than or Equals 1” would trigger this workflow for every scan with 1 or more match results.



Definition:

X Total Matches ▾ | Greater Than or Equals ▾ | 1 +

4. On the ***Endpoints*** tab, select the agent(s) that will execute the Python script.
5. On the ***Actions*** tab, enable ***Perform the following remediation action*** and select “Execute script.”
6. Click the ellipsis button (“…”) and upload the following command as a batch file:

```
python c:\temp\google_drive.py %Location%
```

7. Finalize the creation of this Workflow by clicking ***Finish***.

Creating a Scan Policy

Once the Workflow has been assigned to the Spirion agent(s), Google Drive is ready to be targeted as a repository in a discovery scan. Please reference the documentation under each step for additional guidance.

1. From the ***Policies*** tab, select “Create” from the ***Policy*** pulldown menu.
2. Configure the search’s data types by adjusting its ***Settings***.
    a. Spirion KB Article: Getting Started with Scheduled Tasks

3. Configure the search to target Google Drive as a ***Search Location***.
    a. Spirion KB Article: How to Search Google Drive

Running a Scan

As long as the Workflow and Scheduled Task have been applied to the same Spirion agent(s), running a console-initiated scan will trigger the scripted interactions with Google Drive's API.

1. From the ***Policies*** tab, select the Scheduled Task Policy from the ***Policy List***.
2. Scans can be initiated in one of two ways:
    a. One-off scan:
        i. Right click the policy and select ***Initiate Search*** from the ***Search*** menu.
    b. Scheduled scan:
        i. Navigate to ***Scheduled Tasks*** and click ***Add*** from the ribbon menu.
3. The scan will initiate, searching the Google Drive account(s) specified in the policy created per the previous section.

Outcomes

The accuracy and flexibility of Spirion search logic combined with the dynamic capabilities of its scripting engine readily incorporates external functionality into the selective automation handled by SDM Workflows and SDP Playbooks.

Through Google Drive's API, Spirion enhances the security posture of Google Workspace by leveraging Privacy-Grade data discovery to:

1. Apply Drive labels to standard unstructured file types, such as PDFs, images, and Microsoft Office files.
2. Apply Drive labels to cloud-native content from editors like Docs and Sheets.
3. Audit match locations for internal and external share permissions.

Google Drive Labels

The automated script runs at the end of a Spirion scan, after all locations have been evaluated for matches. Drive labels will be applied automatically at this time.

Spirion Scan Results

Once completed, scan results are returned to the Spirion console. Match locations will include the associated Google Drive user's email address as the root path for each file reported.

Troubleshooting

Spirion Scan Logs

Custom script execution can be verified by auditing Spirion scan logs. In addition to confirming a successful connection to the repository environment targeted in a scan, logs will also indicate that the workflow script has been executed for any given match location:

```
"USER ACTION""Successfully executed script (via workflow) on the file...
```

However, the statement above only indicates that the batch file ran. Testing the Python script manually is necessary if the expected outcome of the Python script is not observed

API Script Logs

The Python script used in this article writes logs to a user-defined location or `C:\temp` by default. Logs are configured to rotate hourly, with success and failure messages confirming the outcome of automated API requests informed by Spirion scan results.

NOTE: Optional logging can be enabled for additional verbosity by leaving ***line 194*** uncommented in the example script.

Manual Python Verification

Console-initiated scans control the Spirion (Windows) agent using the LocalSystem account by default. Since Spirion search logs only indicate if the batch file ran properly, any Python errors will not be reported.

To verify that the Python script executes properly prior to being invoked during a scan, launch a command prompt as the LocalSystem account and call the Python script from that terminal. PsExec provides a convenient means of launching cmd.exe with the appropriate context.

Errors preventing the Python script from completing will be displayed in this terminal. Common causes of concern would be: 1) verifying that the LocalSystem account has access to Python, and 2) ensuring that any module dependencies are also accessible to LocalSystem.