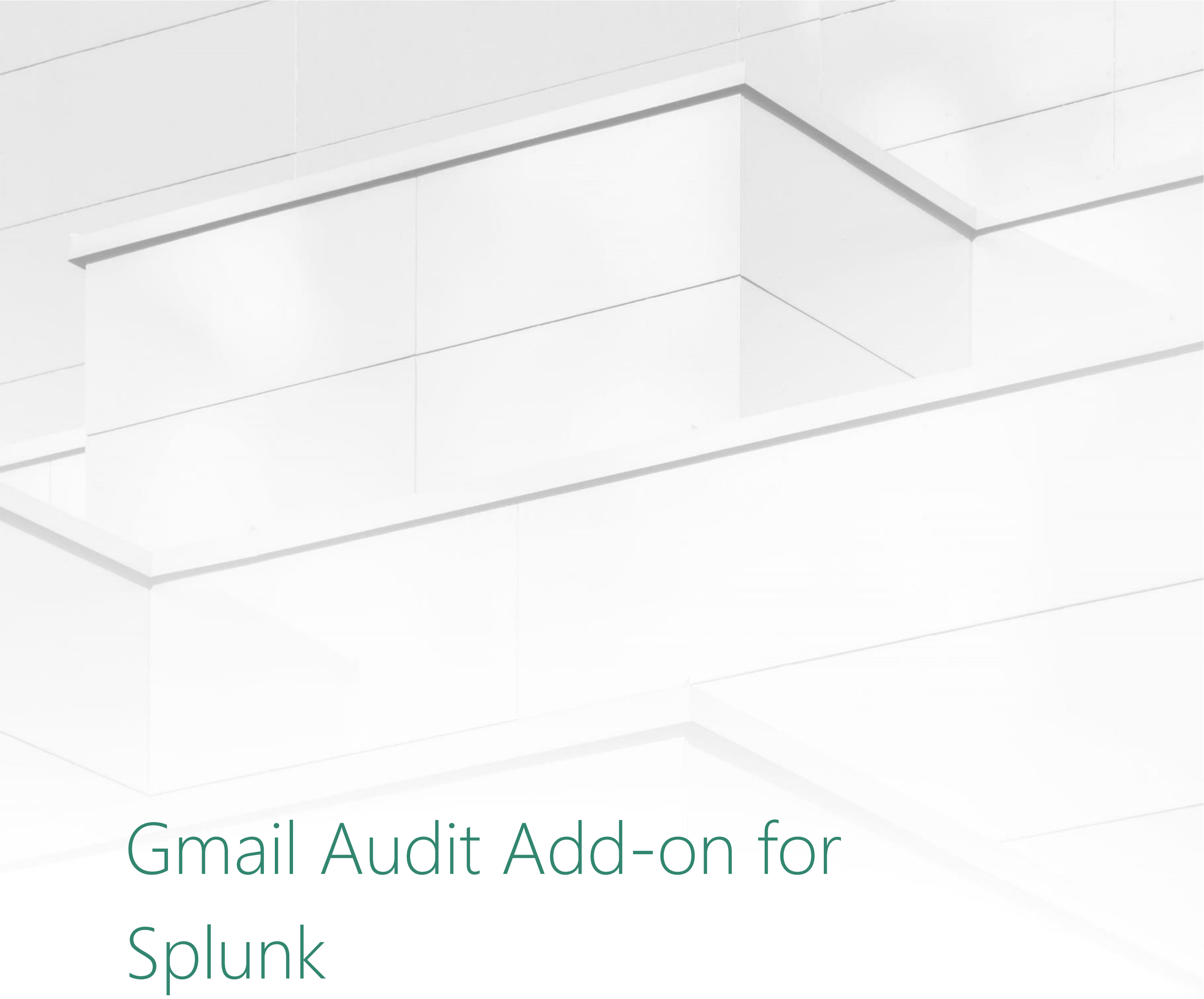


Gmail Audit Add-on for Splunk

Nick von Korff

Nick joined Sourced in 2015 as a Data Visualisation Consultant, specialising in monitoring and data analysis tools, such as Splunk and Datadog. He has worked in IT for over 20 years in a range of industries and roles including systems administration, IVR system programming, network administration, database administration and systems monitoring.

sourcedgroup.com/blog/

sourced
cloud at scale™

Introduction

As part of a cyber-security programme at one of our larger customers, there was a requirement to monitor email header data from one of the client's subsidiary organisations. This data needed to be stored in the Splunk deployment of the parent organisation. While it sounds like a reasonably straightforward requirement, there were a lot of moving parts (many of which needed to be built) in order to achieve this.

Background and Requirements

The subsidiary organisation, unlike the parent organisation, uses Google G Suite and Gmail for their user management, authentication, mail and many of the other features that G Suite provides.

There is an existing Splunk add-on (G Suite for Splunk) for monitoring G Suite environments. This add-on is great and was used to meet a number of other monitoring requirements for the cyber-security programme, but it doesn't do email header monitoring.

One of the other requirements we had was to pull a directory listing of all users and their attributes from the G Suite Directory on a regular basis, and this was another thing that the existing add-on did not support.

Some of the additional cyber-security requirements, beyond simply extracting the available mail headers, were to handle headers for multi-part MIME messages, to extract details (specifically file names and sizes) of any attachments in the message, and finally, to provide a list of any links that were included in the original message.

In order to meet the requirements, we needed a process to accomplish the following three main tasks and to push any data generated into Splunk:

- Retrieve a listing of all users and their attributes from the G Suite Directory;
- Enable auditing on any users that do not already have auditing enabled; and
- Retrieve new audit events and extract relevant information.

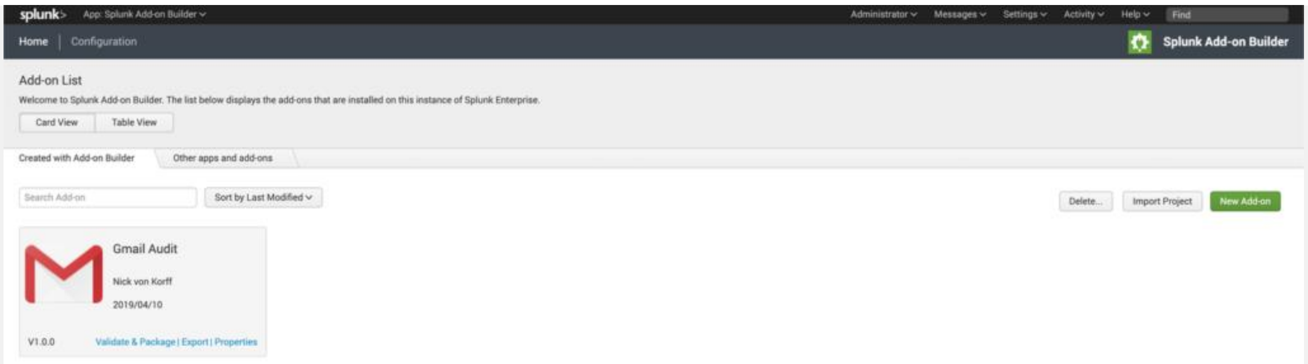
Initially this was done through three separate Python scripts, which were cobbled together into a loosely-formed Splunk add-on. While it worked, the biggest issue with this approach was that the credential management wasn't great, i.e. it relied on a .json file stored on disk holding the credentials, and that it wasn't user-friendly to setup and configure. There was no user interface and any parameters needed to be updated in the scripts themselves.

Splunk Add-on Builder

To solve the credential management and user-friendliness issues, I decided that in my own time, I would build a new add-on from scratch, using the Splunk Add-on Builder and would incorporate the logic from the existing Python scripts into three modular inputs.

The Splunk add-on builder takes care of generating the user interface for the creation of inputs and to accept input parameters so that no modification of the scripts is necessary when using the add-on in other environments. The add-on builder also provides a number of Splunk-specific helper functions to simplify development, and allows developers to run their add-on through a series of validation and pre-certification tests, ensuring that best-practices are followed, naming conventions are adhered to and that the code does not contain any known security vulnerabilities.

In the next few screenshots, I'll walk through the builder process at a high level and show some of its features:

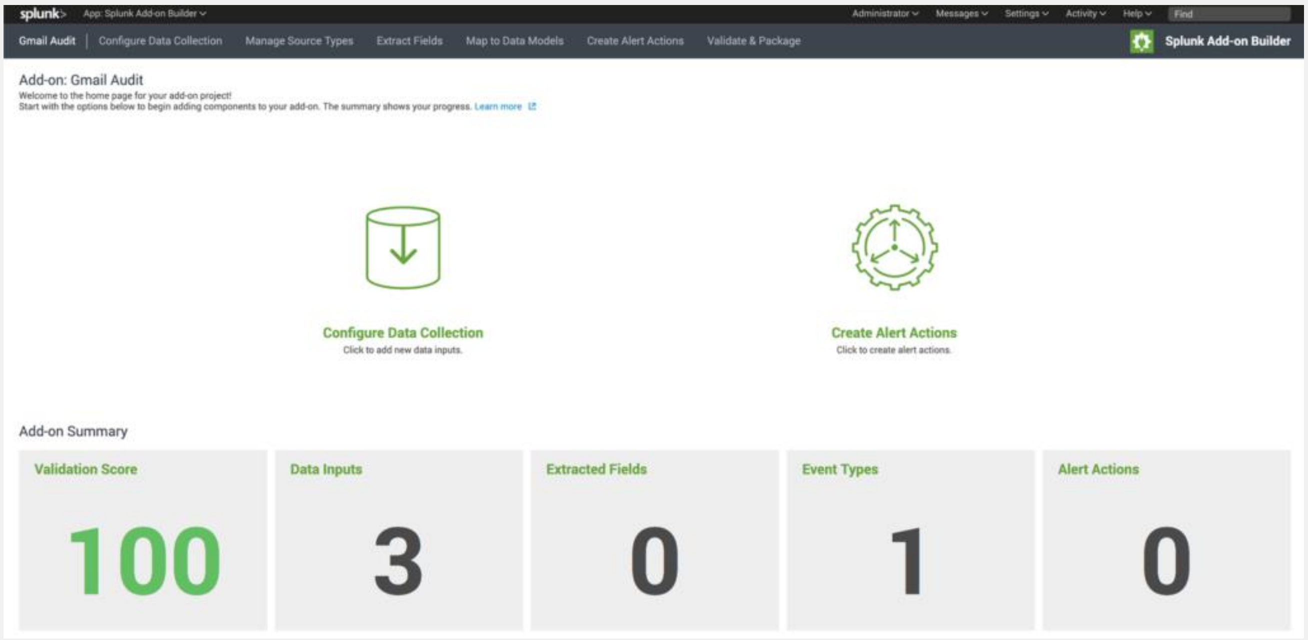


Add-on List

Welcome to Splunk Add-on Builder. The list below displays the add-ons that are installed on this instance of Splunk Enterprise.

Gmail Audit
Nick von Korff
2019/04/10
V1.0.0

Splunk Add-on Builder showing the Gmail Audit app



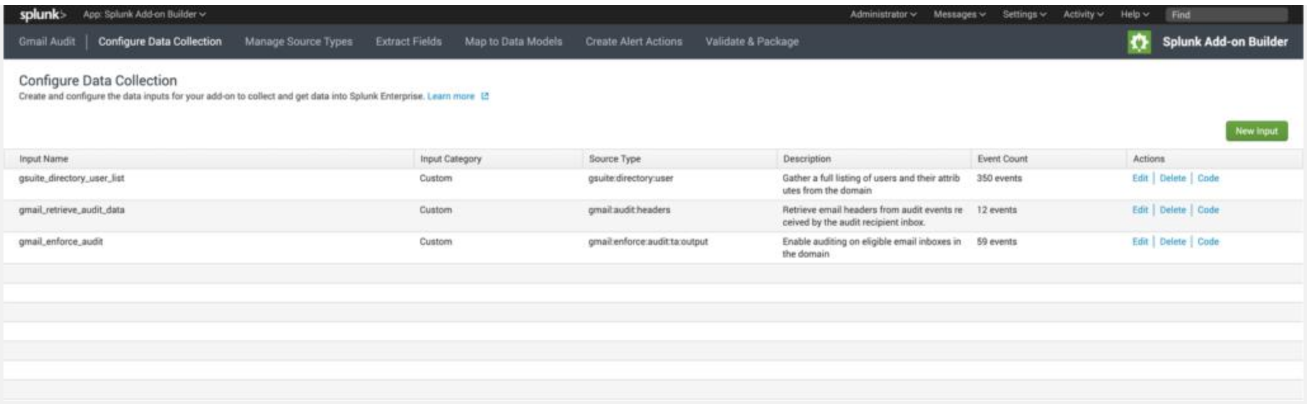
Add-on: Gmail Audit

Welcome to the home page for your add-on project!
Start with the options below to begin adding components to your add-on. The summary shows your progress.

Add-on Summary

Validation Score	Data Inputs	Extracted Fields	Event Types	Alert Actions
100	3	0	1	0

Splunk Add-on Builder overview screen for Gmail Audit app



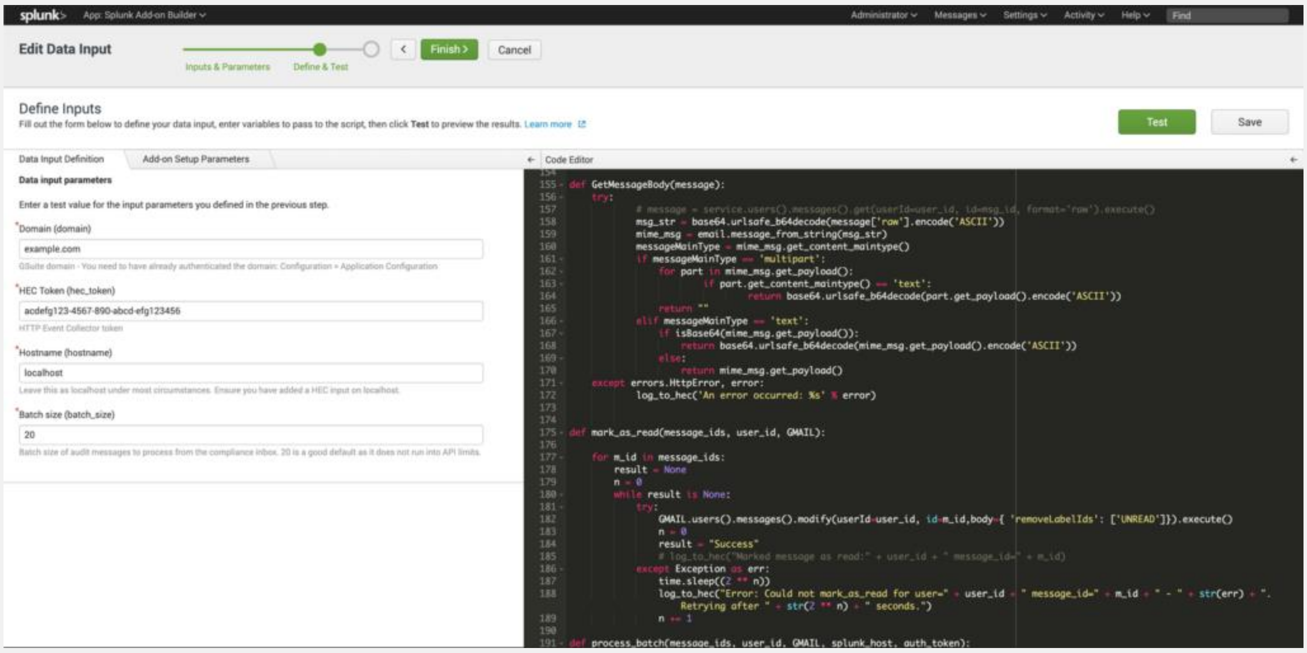
splunk> App: Splunk Add-on Builder
Administrator Messages Settings Activity Help Find
Gmail Audit | Configure Data Collection Manage Source Types Extract Fields Map to Data Models Create Alert Actions Validate & Package
Splunk Add-on Builder
New Input

Configure Data Collection

Create and configure the data inputs for your add-on to collect and get data into Splunk Enterprise. Learn more

Input Name	Input Category	Source Type	Description	Event Count	Actions
gsuite_directory_user_list	Custom	gsuite:directory:user	Gather a full listing of users and their attributes from the domain	350 events	Edit | Delete | Code
gmail_retrieve_audit_data	Custom	gmail:audit:headers	Retrieve email headers from audit events received by the audit recipient inbox.	12 events	Edit | Delete | Code
gmail_enforce_audit	Custom	gmail:enforce:audit:ta:output	Enable auditing on eligible email inboxes in the domain	59 events	Edit | Delete | Code

Splunk Add-on Builder Data Collection screen, showing the three modular inputs configured for Gmail Audit app



Define Inputs

Fill out the form below to define your data input, enter variables to pass to the script, then click **Test** to preview the results. Learn more

Data input parameters

Enter a test value for the input parameters you defined in the previous step.

*Domain (domain): example.com
GSuite domain - You need to have already authenticated the domain: Configuration » Application Configuration

*HEC Token (hec_token): acdefg123-4567-890-abcd-efg123456
HTTP Event Collector token

*Hostname (hostname): localhost
Leave this as localhost under most circumstances. Ensure you have added a HEC input on localhost.

*Batch size (batch_size): 20
Batch size of audit messages to process from the compliance inbox. 20 is a good default as it does not run into API limits.

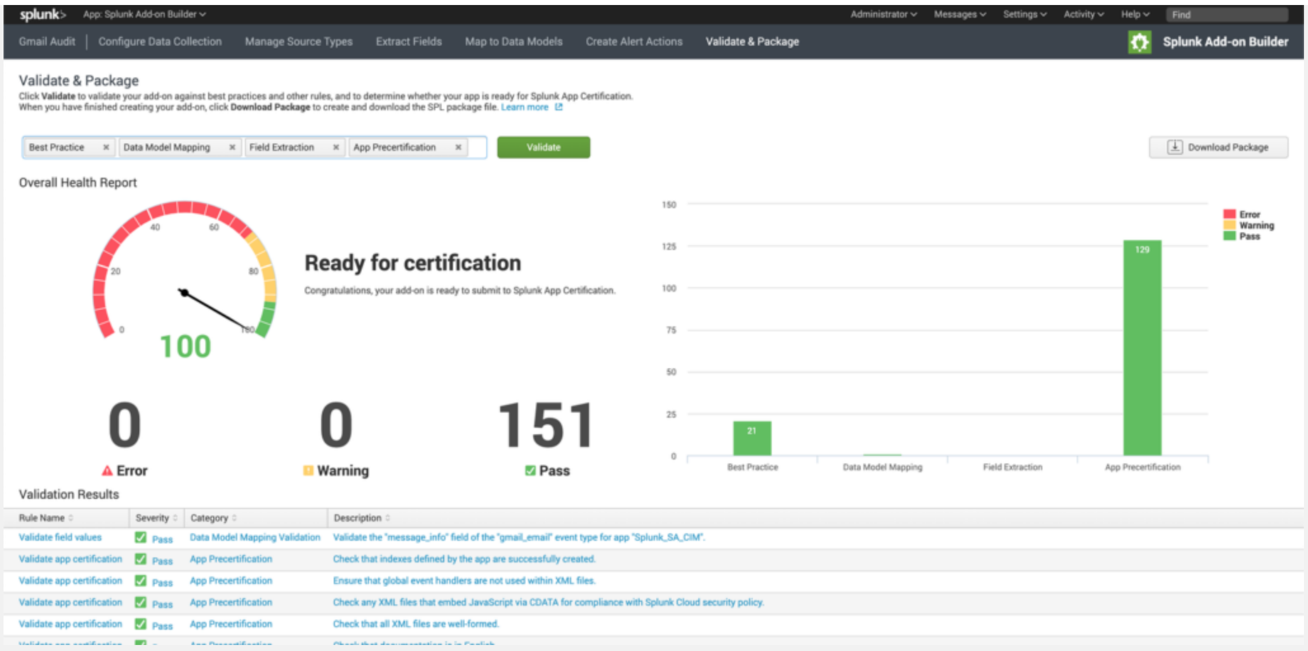
```
def GetMessageBody(message):
    try:
            # message = service.users().messages().get(userId=user_id, id=msg_id, format='raw').execute()
            msg_str = base64.urlsafe_b64decode(message['raw'].encode('ASCII'))
            mime_msg = email.message_from_string(msg_str)
            messageMainType = mime_msg.get_content_maintype()
            if messageMainType == 'multipart':
                for part in mime_msg.get_payload():
                    if part.get_content_maintype() == 'text':
                        return base64.urlsafe_b64decode(part.get_payload().encode('ASCII'))
                return ""
            elif messageMainType == 'text':
                if isBase64(mime_msg.get_payload()):
                    return base64.urlsafe_b64decode(mime_msg.get_payload().encode('ASCII'))
                else:
                    return mime_msg.get_payload()
    except errors.HttpError, error:
            log_to_hec('An error occurred: %s' % error)


def mark_as_read(message_ids, user_id, GMAIL):

    for m_id in message_ids:
        result = None
        n = 0
        while result is None:
            try:
                GMAIL.users().messages().modify(userId=user_id, id=m_id,body={ 'removeLabelIds': ['UNREAD']}).execute()
                n = 0
                result = "Success"
                # log_to_hec("Marked message as read:" + user_id + " message_id=" + m_id)
            except Exception as err:
                time.sleep((2 ** n))
                log_to_hec("Error: Could not mark_as_read for user=" + user_id + " message_id=" + m_id + " - " + str(err) + ".
                    Retrying after " + str(2 ** n) + " seconds.")
                n += 1

def process_batch(message_ids, user_id, GMAIL, splunk_host, auth_token):
```

One of the great features of the Splunk Add-on Builder is the ability to view, edit and test your code for each modular input directly within the app, including populating test input parameters.



splunk> App: Splunk Add-on Builder
Administrator Messages Settings Activity Help Find
Gmail Audit | Configure Data Collection Manage Source Types Extract Fields Map to Data Models Create Alert Actions Validate & Package
Splunk Add-on Builder
Best Practice | Data Model Mapping | Field Extraction | App Precertification
Validate
Download Package
Overall Health Report
100
Ready for certification
0 Error
0 Warning
151 Pass
Error
Warning
Pass
21
129
Best Practice
Data Model Mapping
Field Extraction
App Precertification

Validate & Package

Click **Validate** to validate your add-on against best practices and other rules, and to determine whether your app is ready for Splunk App Certification.
When you have finished creating your add-on, click **Download Package** to create and download the SPL package file. Learn more

Congratulations, your add-on is ready to submit to Splunk App Certification.

Validation Results

Rule Name	Severity	Category	Description
Validate field values	✓ Pass	Data Model Mapping Validation	Validate the "message_info" field of the "gmail_email" event type for app "Splunk_SA_CIM".
Validate app certification	✓ Pass	App Precertification	Check that indexes defined by the app are successfully created.
Validate app certification	✓ Pass	App Precertification	Ensure that global event handlers are not used within XML files.
Validate app certification	✓ Pass	App Precertification	Check any XML files that embed JavaScript via CDATA for compliance with Splunk Cloud security policy.
Validate app certification	✓ Pass	App Precertification	Check that all XML files are well-formed.

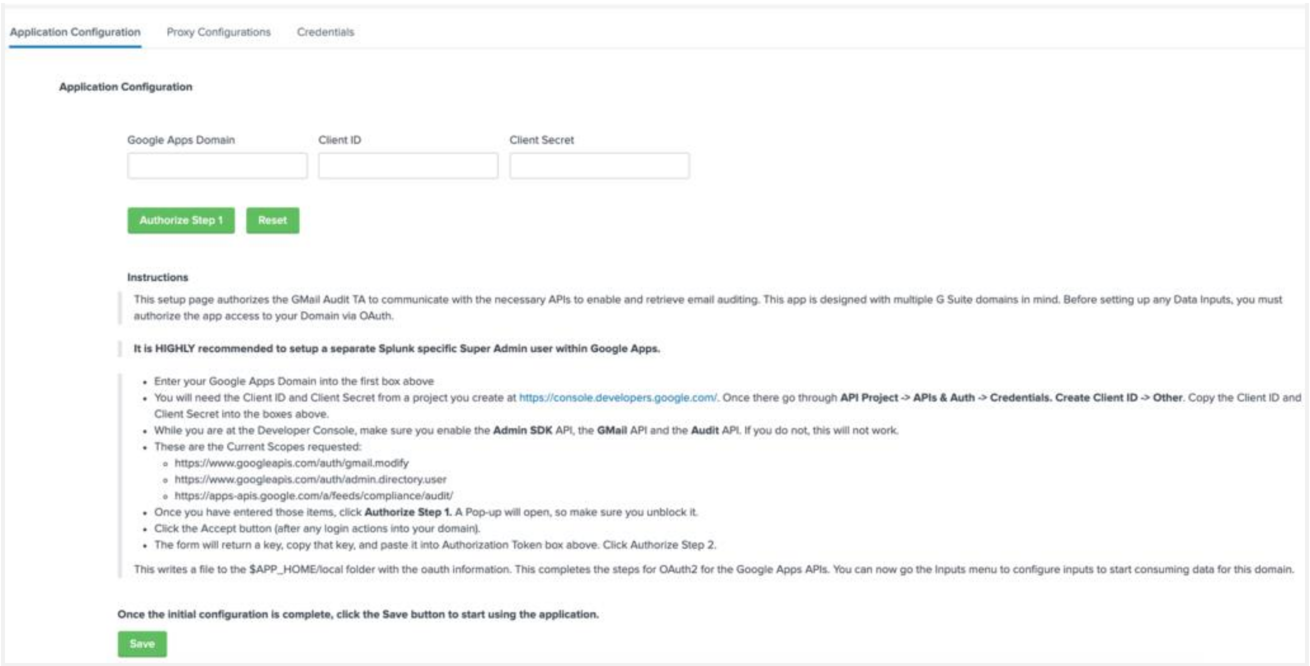
The Validate and Package screen showing the results of the validation checks and pre-certification checks performed by the Splunk Add-on Builder for the Gmail Audit app

Credential Management

I realised that the credential management process in the existing G Suite for Splunk add-on was doing almost exactly what I needed. As a result, if I could incorporate that process into my add-on and have the modular inputs retrieve credentials from the Splunk encrypted credential store, then it would solve both the issue of credentials being stored in a file on disk, and would provide a neat user interface for storing and authorising new sets of credentials. Huge shout-out to Kyle Smith who developed the G Suite for Splunk add-on.

It took some time, but I eventually managed to extract the sections of code from the G Suite for Splunk add-on that handle the input, storage and authorisation of G Suite credentials, incorporate them into my add-on and modify the code to suit my needs.

The fields in the retrieved audit events are mapped to the Email data model of the Splunk Common Information Model (CIM). This means that those events are readily available for premium Splunk applications, like Splunk Enterprise Security, to begin analysing for potential security threats and malicious activity.



Application Configuration

Instructions

This setup page authorizes the GMail Audit TA to communicate with the necessary APIs to enable and retrieve email auditing. This app is designed with multiple G Suite domains in mind. Before setting up any Data Inputs, you must authorize the app access to your Domain via OAuth.

It is HIGHLY recommended to setup a separate Splunk specific Super Admin user within Google Apps.

- Enter your Google Apps Domain into the first box above
- You will need the Client ID and Client Secret from a project you create at https://console.developers.google.com/. Once there go through **API Project -> APIs & Auth -> Credentials. Create Client ID -> Other**. Copy the Client ID and Client Secret into the boxes above.
- While you are at the Developer Console, make sure you enable the **Admin SDK** API, the **GMail** API and the **Audit** API. If you do not, this will not work.
- These are the Current Scopes requested:
  - https://www.googleapis.com/auth/gmail.modify
  - https://www.googleapis.com/auth/admin.directory.user
  - https://apps-apis.google.com/a/feeds/compliance/audit/
- Once you have entered those items, click **Authorize Step 1.** A Pop-up will open, so make sure you unblock it.
- Click the Accept button (after any login actions into your domain).
- The form will return a key, copy that key, and paste it into Authorization Token box above. Click Authorize Step 2.

This writes a file to the $APP_HOME/local folder with the oauth information. This completes the steps for OAuth2 for the Google Apps APIs. You can now go the Inputs menu to configure inputs to start consuming data for this domain.

Once the initial configuration is complete, click the Save button to start using the application.

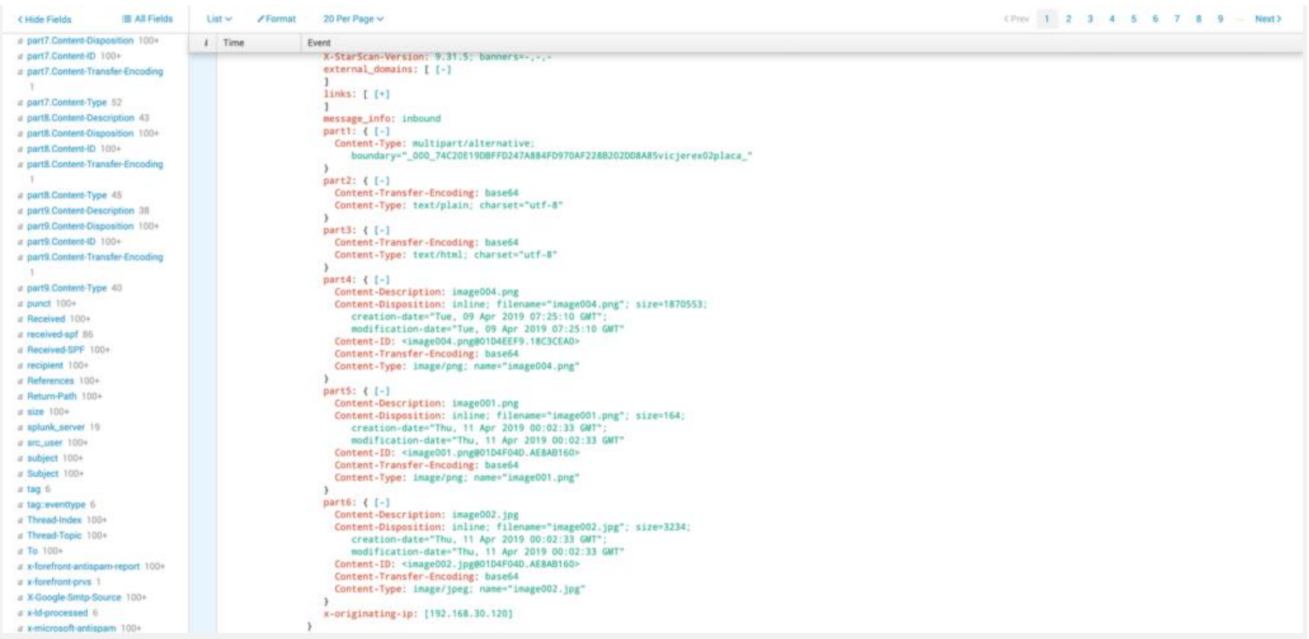
This is the credential management screen which guides you through the 2-step authorisation process. Once the credentials are authorised, the credentials are encrypted and stored within Splunk, ready for the modular inputs to access. This functionality is duplicated almost exactly from the G Suite for Splunk add-on. Thanks again to Kyle Smith for developing that add-on.

Permission

Qantas Airways commissioned the original development of much of this code. We sought their permission to release this add-on to the publicly accessible Splunkbase, which they have kindly given. This means that any bugs found/fixed and any newly developed features will be available to them (and the wider community) at no further cost.

Example Data

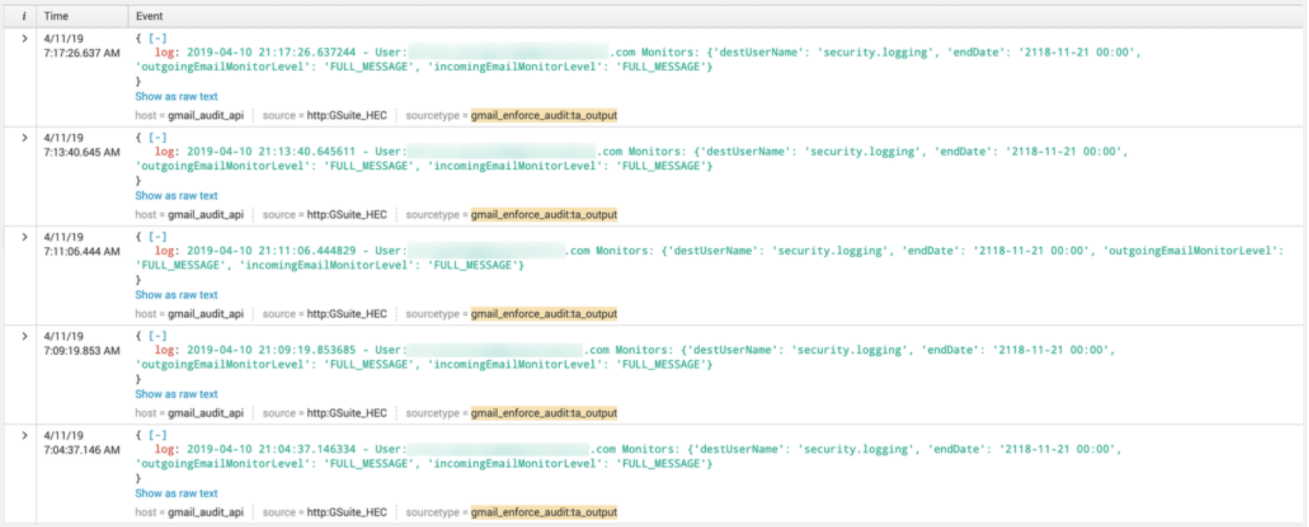
The add-on requires a user to be setup within G Suite with appropriate permissions to enable auditing on other users and to read the G Suite Directory. The email inbox of this user is also used as the “compliance inbox” to which audited emails are delivered. When it comes time to retrieve audit events, the script authenticates as that user, reads any unread messages from “compliance-noreply@google.com”, marks processed messages as read and finally loads the data into Splunk.



```
a part7.Content-Disposition 100+
a part7.Content-ID 100+
a part7.Content-Transfer-Encoding
  1
a part7.Content-Type 52
a part8.Content-Description 43
a part8.Content-Disposition 100+
a part8.Content-ID 100+
a part8.Content-Transfer-Encoding
  1
a part8.Content-Type 45
a part9.Content-Description 38
a part9.Content-Disposition 100+
a part9.Content-ID 100+
a part9.Content-Transfer-Encoding
  1
a part9.Content-Type 40
a punct 100+
a Received 100+
a received-spf 86
a Received-SPF 100+
a recipient 100+
a References 100+
a Return-Path 100+
a size 100+
a splunk_server 19
a src_user 100+
a subject 100+
a Subject 100+
a tag 6
a tag::eventtype 6
a Thread-Index 100+
a Thread-Topic 100+
a To 100+
a x-forefront-antispam-report 100+
a x-forefront-prvs 1
a X-Google-Smtp-Source 100+
a x-ld-processed 6
a x-microsoft-antispam 100+
```

```
   X-StarScan-Version: 9.31.5; banners=-,-,-
   external_domains: [ [-]
   ]
   links: [ [+]
   ]
   message_info: inbound
   part1: { [-]
     Content-Type: multipart/alternative;
        boundary="_000_74C20E19DBFFD247A884FD970AF228B202D08A85vicjerex02placa_"
   }
   part2: { [-]
     Content-Transfer-Encoding: base64
     Content-Type: text/plain; charset="utf-8"
   }
   part3: { [-]
     Content-Transfer-Encoding: base64
     Content-Type: text/html; charset="utf-8"
   }
   part4: { [-]
     Content-Description: image004.png
     Content-Disposition: inline; filename="image004.png"; size=1870553;
        creation-date="Tue, 09 Apr 2019 07:25:10 GMT";
        modification-date="Tue, 09 Apr 2019 07:25:10 GMT"
     Content-ID: <image004.png@01D4EEF9.18C3CEA0>
     Content-Transfer-Encoding: base64
     Content-Type: image/png; name="image004.png"
   }
   part5: { [-]
     Content-Description: image001.png
     Content-Disposition: inline; filename="image001.png"; size=164;
        creation-date="Thu, 11 Apr 2019 00:02:33 GMT";
        modification-date="Thu, 11 Apr 2019 00:02:33 GMT"
     Content-ID: <image001.png@01D4F04D.AE8AB160>
     Content-Transfer-Encoding: base64
     Content-Type: image/png; name="image001.png"
   }
   part6: { [-]
     Content-Description: image002.jpg
     Content-Disposition: inline; filename="image002.jpg"; size=3234;
        creation-date="Thu, 11 Apr 2019 00:02:33 GMT";
        modification-date="Thu, 11 Apr 2019 00:02:33 GMT"
     Content-ID: <image002.jpg@01D4F04D.AE8AB160>
     Content-Transfer-Encoding: base64
     Content-Type: image/jpeg; name="image002.jpg"
   }
   x-originating-ip: [192.168.30.120]
}
```

An example of some of the mail headers being captured by the add-on. You can see that each part of a multi-part MIME message gets its own "part x" section. Attachments contain size and filename information (where this data is available) and there is a list of links (not expanded for privacy reasons) contained in the original email.



```
>  4/11/19          { [-]
   7:17:26.637 AM     log: 2019-04-10 21:17:26.637244 - User: [redacted] .com Monitors: {'destUserName': 'security.logging', 'endDate': '2118-11-21 00:00',
                    'outgoingEmailMonitorLevel': 'FULL_MESSAGE', 'incomingEmailMonitorLevel': 'FULL_MESSAGE'}
                    }
                    Show as raw text
                    host = gmail_audit_api   source = http:GSuite_HEC   sourcetype = gmail_enforce_audit:ta_output

>  4/11/19          { [-]
   7:13:40.645 AM     log: 2019-04-10 21:13:40.645611 - User: [redacted] .com Monitors: {'destUserName': 'security.logging', 'endDate': '2118-11-21 00:00',
                    'outgoingEmailMonitorLevel': 'FULL_MESSAGE', 'incomingEmailMonitorLevel': 'FULL_MESSAGE'}
                    }
                    Show as raw text
                    host = gmail_audit_api   source = http:GSuite_HEC   sourcetype = gmail_enforce_audit:ta_output

>  4/11/19          { [-]
   7:11:06.444 AM     log: 2019-04-10 21:11:06.444829 - User: [redacted] .com Monitors: {'destUserName': 'security.logging', 'endDate': '2118-11-21 00:00', 'outgoingEmailMonitorLevel':
                    'FULL_MESSAGE', 'incomingEmailMonitorLevel': 'FULL_MESSAGE'}
                    }
                    Show as raw text
                    host = gmail_audit_api   source = http:GSuite_HEC   sourcetype = gmail_enforce_audit:ta_output

>  4/11/19          { [-]
   7:09:19.853 AM     log: 2019-04-10 21:09:19.853685 - User: [redacted] .com Monitors: {'destUserName': 'security.logging', 'endDate': '2118-11-21 00:00',
                    'outgoingEmailMonitorLevel': 'FULL_MESSAGE', 'incomingEmailMonitorLevel': 'FULL_MESSAGE'}
                    }
                    Show as raw text
                    host = gmail_audit_api   source = http:GSuite_HEC   sourcetype = gmail_enforce_audit:ta_output

>  4/11/19          { [-]
   7:04:37.146 AM     log: 2019-04-10 21:04:37.146334 - User: [redacted] .com Monitors: {'destUserName': 'security.logging', 'endDate': '2118-11-21 00:00',
                    'outgoingEmailMonitorLevel': 'FULL_MESSAGE', 'incomingEmailMonitorLevel': 'FULL_MESSAGE'}
                    }
                    Show as raw text
                    host = gmail_audit_api   source = http:GSuite_HEC   sourcetype = gmail_enforce_audit:ta_output
```

Example events from auditing being enabled on users in the domain. FULL_MESSAGE monitoring is enabled for both incoming and outgoing email, with a recipient address of "security.logging". Note that if multiple domains are being managed, that the "security.logging" user needs an alias in each domain. An enddate is required, which has been set to 100 years in the future.

Benefits

Having the ability to automate retrieval of Gmail audit events in this manner alleviates the need for operators to assume high levels of privilege into G Suite, such as SuperAdmin, for performing this task. A single set of credentials created specifically for this purpose means that access to those credentials can be locked down and monitored.

For companies that are broad users of G Suite, including GCP, this reduces the risk associated with managing G Suite environments as you can now keep SuperAdmin far more locked down, in that you don't need to provide this level of access to one or more administrator accounts for this purpose.

Access to the data stored in Splunk can be tightly controlled using Splunk's fine-grained role-based user access controls.

The fields in the retrieved audit events are mapped to the Email data model of the Splunk Common Information Model (CIM). This means that those events are readily available for premium Splunk applications, like Splunk Enterprise Security, to begin analysing for potential security threats and malicious activity.

Conclusion

Using the Splunk add-on builder, some Python knowledge and the credential management process from the G Suite for Splunk add-on, I was able to:

- Improve upon the initial incantation of the Gmail audit process;
- Make the add-on more secure;
- Make the add-on more user-friendly; and
- Publish my first Splunk add-on to Splunkbase.

The add-on is available for download from Splunkbase here: Gmail Audit