Using audit logs to track activity

ON THIS PAGE

What you need
List the account setting changes within a specific time period
Introduction
How it works
Example output
Get information about a user
How it works
Example output

Zendesk audit logs are detailed records that track the changes made within a Zendesk account. These logs capture information such as who made the changes, what was changed, and when the changes occurred. They are critical for monitoring administrative actions, ensuring compliance, and troubleshooting issues.

This article gives two examples of using the Zendesk Audit Log APIs to monitor account setting changes within a specified date range, and retrieve information about the user responsible for those changes.

What you need

To use Zendesk audit logs, you'll need a Zendesk account. You'll also need to be assigned the role of admin in the account.

To run this script, you'll need Python and the requests and arrow third-party libraries.

List the account setting changes within a specific time period

Introduction

In this Python example, we'll demonstrate how to use the Audit Logs API to search for account setting updates within a specified date range, and convert the timestamps of these updates to a local time zone for better readability.

```python
import os
import json

import requests
import arrow

# Zendesk API credentials stored as environment variables
# for security reasons
ZENDESK_API_TOKEN = os.getenv('ZENDESK_API_TOKEN')
ZENDESK_USER_EMAIL = os.getenv('ZENDESK_USER_EMAIL')
ZENDESK_SUBDOMAIN = os.getenv('ZENDESK_SUBDOMAIN')

AUTH = f'{ZENDESK_USER_EMAIL}/token', ZENDESK_API_TOKEN

# Function to search for account_setting updates
def search_account_setting_updates(start_date, end_date):
    audit_logs: list = []
    source_type = "account_setting"
    try:
        # Construct the API URL
        url = f"https://{ZENDESK_SUBDOMAIN}.zendesk.com/api/v2/audit_logs.json"

        # Define query parameters
        params = {
            'filter[source_type]': source_type,
            'filter[created_at][]': [
                start_date,
                end_date
             ]
        }

        # Make the API request
        response = requests.get(url, params=params, auth=AUTH)

        # Paginate through the results
        while url:
            # Check if the request was successful
            if response.status_code == 200:
                page = response.json()
                audit_logs.extend(page['audit_logs'])
                if page['next_page']:
                    url = page['next_page']
                    response = requests.get(url, auth=AUTH)
                    page = response.json()
                    audit_logs.extend(page['audit_logs'])
                else:
                    url = ''
            else:
                print(f"Error: {response.status_code} - {response.text}")
                return None
        return audit_logs
    except Exception as e:
        print(f"An error occurred: {e}")
        return None

def get_date_range_from_user():
    # Prompts for the start and end dates
    start_date_input = input("Enter the start date (YYYY-MM-DD): ")
    end_date_input = input("Enter the end date (YYYY-MM-DD): ")
    # Quick conversions to UTC format in ISO 8601
    start_date = f'{start_date_input}T00:00:00Z'
    end_date = f'{end_date_input}T23:59:59Z'

    return {'start_date': start_date, 'end_date': end_date}

# Main program
def main():
    # Get the starting and ending dates for the search
    date_range = get_date_range_from_user()
    # Get the setting updates with the API
    audit_logs = search_account_setting_updates(date_range['start_date'], date_range['end_date'])
    if audit_logs:
        print("\n\nAccount setting updates found:")
        for log in audit_logs:
            person = log['actor_name']
            description = log['change_description']
            local_datetime = arrow.get(log['created_at']).to('local').format("YY-MM-DD HH:mm:ss")
            source_type = log['source_label']
            actor_id = log['actor_id']
            print(
                f"ID: {log['id']}, At {local_datetime} User: {person} with id: {actor_id} updated '{source_type}': {description}"
            )
    else:
        print("No account setting updates found.")
if __name__ == "__main__":
    main()
```

How it works

The script does the following:

1. Imports the required libraries.

   - `requests` for making HTTP requests.
   - `arrow` for date and time manipulations.
   - `json` for formatting JSON data.
   - `os` for accessing the environment variables.

2. Retrieves the Zendesk API credentials (`ZENDESK_API_TOKEN`, `ZENDESK_USER_EMAIL`) and subdomain (`ZENDESK_SUBDOMAIN`) from environment variables.

   Environment variables are stored outside the source code, reducing the risk of exposing sensitive information in version control systems. This isolation helps prevent credentials from being accidentally committed to public repositories.
3. Sets up HTTP authentication using the Zendesk email and API token.

   ```
   AUTH = f'{ZENDESK_USER_EMAIL}/token', ZENDESK_API_TOKEN
   ```

4. Creates the `search_account_setting_updates` function to construct an API URL using the provided subdomain. See Show Audit Log.

   It then filters the audit log based on the provided `source_type` and the `created_at date` range between `start_date` and `end_date`.

   ```
   params = {
         'filter[source_type]': source_type,
         'filter[created_at][]': [
             start_date,
             end_date
         ]
     }
   ```

   Afterwards, it makes an authenticated GET request to retrieve the audit logs. If the request is not successful, an error message is displayed.
5. Calls the main function to prompt the user to input the start and end dates in YYYY-MM-DD format.

   It then calls the `search_account_settings_updates` function with the converted date range. If `audit_logs` is not empty, it iterates through each audit log and extracts the following information and prints it to the console:

   - `person` is the name of the user who performed the action.

- `description` is the description of the action.
- `local_datetime` converts the `created_at` timestamp from UTC to the specified local timezone.
- `source_type` is the type or label of the source.
- `actor_id` is the id of the user who performed the action.

Example output

```
Account setting updates found:
ID: 23649114690839, At 2024-05-21 15:34:33 User: Joe Agent with id: 1509756491122
 updated 'Security: Session expiration': Changed from 10 minutes to 30 minutes
ID: 23639756399639, At 2024-05-21 10:36:03 User: Joe Agent with id: 1509756491122
 updated 'Security: Session expiration': Changed from 0 minutes to 10 minutes
ID: 14041457644311, At 2024-05-21 10:20:19 User: Joe Agent with id: 1509756491122
 updated 'End users: Allow users to view and edit their profile data': Turned on
ID: 1501160646542, At 2024-05-21 8:15:12 User: Joe Agent with id: 1509756491122
 updated 'End users: Anybody can submit tickets': Turned on
```

This example extracts specific information about the changes. If you want all the information that audit log has about that activity, update the main program as follows:

```
if audit_logs:
    print("\n\nAccount setting updates found:")
    for log in audit_logs:
        print(json.dumps(log, indent=2))
else:
    print("No account setting updates found.")
```

Here's an example output:

```
Account setting updates found:
{
  "url": "https://abc.zendesk.com/api/v2/audit_logs/23649114690839.json",
  "id": 23649114690839,
  "action_label": "Updated",
  "actor_id": 1509756491122,
  "source_id": 23639740139415,
  "source_type": "account_setting",
  "source_label": "Security: Session expiration",
  "action": "update",
  "change_description": "Changed from 10 minutes to 30 minutes",
  "ip_address": "52.40.123.85",
  "created_at": "2024-05-21T22:34:33Z",
  "actor_name": "Joe Agent"
}
```

```
...
```

Get information about a user

The following Python script uses the Zendesk API to fetch detailed user information and the user's last 20 activities from the audit logs. You can use this script with List the account setting changes within a specific time period to get more information about a user who made account setting changes.

```python
import os
import json
import requests

# Zendesk API credentials stored as environment variables
# for security reasons
ZENDESK_API_TOKEN = os.getenv('ZENDESK_API_TOKEN')
ZENDESK_USER_EMAIL = os.getenv('ZENDESK_USER_EMAIL')
ZENDESK_SUBDOMAIN = os.getenv('ZENDESK_SUBDOMAIN')

AUTH = f'{ZENDESK_USER_EMAIL}/token', ZENDESK_API_TOKEN

# Function to fetch user details
def get_user_details(user_id):
    url = f'https://{ZENDESK_SUBDOMAIN}.zendesk.com/api/v2/users/{user_id}.json'

    response = requests.get(url, auth=AUTH)
    if response.status_code == 200:
        return response.json()
    else:
        print(f'Error fetching user details: {response.status_code}')
        print(response.text)
        return None

# Function to fetch user's last 20 activities
def get_user_activities(user_id):
    url = f'https://{ZENDESK_SUBDOMAIN}.zendesk.com/api/v2/audit_logs.json'

    params = {
        'filter[user_id]': user_id,
        'per_page': 20,
        'sort_by': 'created_at',
        'sort_order': 'desc'
    }
    response = requests.get(url, auth=AUTH, params=params)
    if response.status_code == 200:
        return response.json()
    else:
        print(f'Error fetching user activities: {response.status_code}')
        print(response.text)
        return None

def main():
    # Prompt for the user id
    user_id = input("Enter the user id: ")
    # Fetch user details
    user_details = get_user_details(user_id)
    if user_details:
        print('User Details:')
        print(json.dumps(user_details, indent=2))
    # Fetch user activities
    user_activities = get_user_activities(user_id)
    if user_activities:
        print('User Activities (last 20):')
        print(json.dumps(user_activities, indent=2))
if __name__ == "__main__":
    main()
```

How it works

The script does the following:

1. Imports the required libraries.

   - `requests` for making HTTP requests.
   - `json` for formatting JSON data.
   - `os` for accessing the environment variables.

2. Retrieves the Zendesk API credentials (`ZENDESK_API_TOKEN`, `ZENDESK_USER_EMAIL`) and subdomain (`ZENDESK_SUBDOMAIN`) from environment variables.
3. Creates a `get_user_details` function that constructs an API URL using the provided subdomain and user id. See Show User.

   It then makes an authenticated GET request to retrieve user details and prints an error message if the request fails.
4. Creates a `get_user_activities` function that constructs an API URL and includes query parameters to filter activities by user id. See List Audit Logs.

   It then specifies parameters for pagination and sorting to fetch the last 20 activities. It uses `params` to:

   - Filter only the records related to the specified user id.
   - Limit the response to 20 records per page.

- Sort the results by the `created_at` field in descending order, showing the most recent records first.

```
params = {
   'filter[user_id]': user_id,
   'per_page': 20,
   'sort_by': 'created_at',
   'sort_order': 'desc'
 }
```

Afterwards, it makes an authenticated GET request to retrieve the user activities and prints an error message if the request fails.

5. Calls the main function to prompt for the user id of the person you want more information.

   It then calls `get_user_details` and `get_user_activities` and displays the user details and last 20 activities in JSON format.

Example output

```
User Details:
{
  "user": {
    "id": 1509756491122,
    "url": "https://abc.zendesk.com/api/v2/users/1509756491122.json",
    "name": "Joe Agent",
    "email": "joe.agent@zendesk.com",
    "created_at": "2021-06-01T14:54:25Z",
    "updated_at": "2024-05-29T15:31:25Z",
    "time_zone": "Pacific Time (US & Canada)",
    "iana_time_zone": "America/Los_Angeles",
    "phone": null,
    "shared_phone_number": null,
    "photo": null,
    "locale_id": 1,
    "locale": "en-US",
    "organization_id": 1500434258322,
    "role": "admin",
    "verified": true,
    "external_id": null,
    "tags": [],
    "alias": null,
    "active": true,
    "shared": false,
    "shared_agent": false,
    "last_login_at": "2024-05-29T15:31:16Z",
    "two_factor_auth_enabled": null,
    "signature": null,
    "details": null,
    "notes": null,
    "role_type": 4,
    "custom_role_id": 1500008664581,
    "moderator": true,
    "ticket_restriction": null,
    "only_private_comments": false,
    "restricted_agent": false,
    "suspended": false,
    "default_group_id": 1500003172542,
    "report_csv": true,
    "user_fields": {
      "social_messaging_user_info": null,
      "whatsapp": null
    }
  }
}

User Activities (last 20):
{
  "audit_logs": [
    {
      "url": "https://abc.zendesk.com/api/v2/audit_logs/23814867857303.json",
      "id": 23814867857303,
      "action_label": "Signed in",
      "actor_id": 1509756491122,
      "source_id": 1509756491122,
      "source_type": "user",
      "source_label": "Team member: Joe Agent",
      "action": "login",
      "change_description":
"Successful sign-in using Zendesk password from
https://abc.zendesk.com/access/login"
,
      "ip_address": "52.40.158.85",
      "created_at": "2024-05-29T15:31:15Z",
      "actor_name": "Joe Agent"
    },
...
```

Join our developer community

Forum Blog Slack