Summary

This document contains threat hunting guidance and queries for detecting abnormal and malicious activity across Snowflake customer database instances. Default retention policies for the relevant views enable threat hunting across the past 1 year (365 days).

Created: June 17, 2024
# Table of Contents

Table of Contents......................................................................................................................................................... 2  
Background........................................................................................................................................................................... 3  
General Snowflake Tips & Tricks........................................................................................................................................... 4  
IAM Review.............................................................................................................................................................................. 6  
  Roles and Permissions Changes.................................................................................................................................................. 6  
  Abnormal Table and DB Access................................................................................................................................................... 8  
  User Creation and Deletion.......................................................................................................................................................... 13  
Query Analysis............................................................................................................................................................................. 15  
  Frequency Analysis........................................................................................................................................................................ 15  
  Error Rate Analysis....................................................................................................................................................................... 19  
  High Resource Consumption......................................................................................................................................................... 26  
  Long Running Queries.................................................................................................................................................................. 33  
  Multi-Day Duplicate Queries........................................................................................................................................................ 36  
Staging/Exfil..................................................................................................................................................................................... 39  
  Data Compression.......................................................................................................................................................................... 39  
  Data Staging.................................................................................................................................................................................... 43  
  Data Streams Outbound................................................................................................................................................................. 47  
  Manual Data Retrieval.................................................................................................................................................................. 50  
Network Analysis............................................................................................................................................................................ 54  
  Network Traffic Spikes................................................................................................................................................................. 54  
User Analysis.................................................................................................................................................................................. 56  
  Abnormal Application Names......................................................................................................................................................... 56  
  Average User Sessions................................................................................................................................................................. 60  
Query for Temporary Stages and External URLs.................................................................................................................... 64  

© Copyright 2024 Mandiant
**Background**

On June 10, 2024, Mandiant released a report on a threat campaign targeting Snowflake customer database instances with the intent of data theft and extortion. Snowflake is a multi-cloud data warehousing platform used to store and analyze large amounts of structured and unstructured data. Mandiant tracks this cluster of activity as UNC5537, a financially motivated threat actor suspected to have stolen a significant volume of records from Snowflake customer environments. UNC5537 is systematically compromising Snowflake customer instances using stolen customer credentials, advertising victim data for sale on cybercrime forums, and attempting to extort many of the victims.

Mandiant's investigation has not found any evidence to suggest that unauthorized access to Snowflake customer accounts stemmed from a breach of Snowflake's enterprise environment. Instead, every incident Mandiant responded to associated with this campaign was traced back to compromised customer credentials.

This guide provides hunting guidance to identify both activity associated with this campaign and other general malicious activity.

Read our [incident response investigation blog post](#) for more information.
General Snowflake Tips & Tricks

Description
This section contains a few tricks that can help keep your threat hunting queries readable, and avoid some issues with parsing and exporting results.

1: Common Table Expressions (CTEs)
A CTE takes the results of a query and stores them as a value which can be referenced in your main query. This can help keep the main body of your query cleaner and easier to edit in the future.

Query
```
WITH sq AS ( //sub query
    SELECT <THINGS>
    FROM <PLACE>
    WHERE <CONDITION>
    GROUP BY <THING-1>
)
SELECT //main query
    <THINGS>
FROM <PLACE> p
    JOIN sq ON sq.<THING-1> = p.<THING-1>
WHERE <CONDITION>
```

2: Timestamp normalization
Timestamps, even when cast to UTC, will only appear normalized in the snowflake UI. Once you export them, they will return to their original timezone. To sidestep this, encapsulate timezones in both CONVERT_TIMEZONE, then TO_VARCHAR, which stores them as immutable string values.

Query
```
TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyymmdd
hh24:mi:ss') AS EARLIEST.UTC,
```
3: JSON extraction

Two main methods exist for JSON extraction:

1. Conversion of the entire JSON blob
2. Text extraction of single fields within a JSON blob

Query

Unset

// blob conversion
PARSE_JSON(DB.SCHEMA.TABLE.JSON_BLOB_FIELD) AS PARSED_JSON
// access nested members
PARSED_JSON:<NESTED FIELD NAME>::STRING AS NESTED_FIELD

Query

Unset

// individual path extraction
JSON_EXTRACT_PATH_TEXT(DB.SCHEMA.TABLE.JSON_BLOB_FIELD, '<NESTED FIELD IN JSON BLOB') AS <NEW_FIELD_NAME>
IAM Review

Roles and Permissions Changes

Description
Hunt for any changes made to roles or permissions that could be associated with attacker activity. This includes abnormal usage of the GRANT statement to:

- Enumerate resources the current user has access to
- Grant additional rights to a user enabling additional access to commands or resources

Observations
During Mandiant's investigation of UNC5537, the SHOW GRANT command was utilized to enumerate permissions and identify what tables they had access to.

Potential Attacker Trends
We observed multiple instances of exfiltration preceded by a large number of "SHOW GRANT" queries, explicitly calling individual usernames. Prior to table selection, the GRANT statement was queried to confirm the current user account had proper access to the table of interest.

Stacking Field Definition

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT_IP / IP</td>
<td>IP address running queries</td>
</tr>
<tr>
<td>USERNAME</td>
<td>Name of the user account that ran the query</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>The name of the application that ran queries, from CLIENT_ENVIRONMENT</td>
</tr>
<tr>
<td>EARLIEST_UTC</td>
<td>First time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>LATEST_UTC</td>
<td>Last time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>UNIQUE_QUERIES</td>
<td>Count of unique queries per grouped variation</td>
</tr>
<tr>
<td>TOTAL_QUERIES</td>
<td>Count of total queries per grouped variation</td>
</tr>
<tr>
<td>DAY_COUNT</td>
<td>Count of unique days active per grouped variation</td>
</tr>
<tr>
<td>UNIQ_USER</td>
<td>Count of unique users per grouped variation</td>
</tr>
<tr>
<td>ARR_UNIQ_USER</td>
<td>Array of users that ran queries per grouped variation</td>
</tr>
<tr>
<td>UNIQ_APP_NAME</td>
<td>Count of unique application names per grouped variation</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td>ARR_APP_NAME</td>
<td>Array of application names per grouped variation</td>
</tr>
<tr>
<td>UNIQ_OS</td>
<td>Count of unique operating systems per grouped variation</td>
</tr>
<tr>
<td>ARR_OP_SYS</td>
<td>Array of operating systems per grouped variation</td>
</tr>
<tr>
<td>UNIQ_IP</td>
<td>Count of unique IPs per grouped variation</td>
</tr>
</tbody>
</table>

Query 1: Spikes in Admin Permission Changes/Views by User, Application Name, IP, and Operating System per Day

Query

```
SELECT
    DATE(START_TIME) AS QUERY_DATE,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)), 'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)), 'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS UNIQ_OP_SYS,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS,
    COUNT(DISTINCT l.CLIENT_IP) AS UNIQ_IP,
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = s.LOGIN_EVENT_ID
WHERE (
    upper(QUERY_TEXT) LIKE '%CREATE USER%' OR
    upper(QUERY_TEXT) LIKE '%ALTER USER%' OR
    upper(QUERY_TEXT) LIKE '%ALTER ACCOUNT%' OR
    upper(QUERY_TEXT) LIKE '%ALTER PASSWORD POLICY%' OR
```
Abnormal Table and DB Access

Description
Most accounts regularly access a limited set of databases, schemas, views, and tables, but unless there is prior insight, an attacker needs to enumerate a multitude of sources. These spikes in the number of databases/schemas/views/tables can be indicative of attacker activity.

Observations
We noticed a very distinctive spike in unique views/tables accessed by user accounts impacted by the attacker. Since there are more views/tables than schemas, and more schemas than databases, analysis of views/tables can show more distinctive peaks and valleys.

Warnings about FP/Noise
Normalizing the results as a fraction from the average can be a better way to look for spikes, rather than the raw aggregate numbers. Raw numbers can cause a loud service account to drown out the enumeration of an attacker’s activity. The below aggregation normalizes the daily unique numbers as a percentage of that entry’s average.

Note: The following queries are stacked by username and application name; however, stacking by IP is explicitly excluded. Since most client environments include a very high number of IP hits, and this query when stacked by date/DB/schema/table can get quite lengthy, an additional
factor of IP causes the number of results to climb to an unmanageable volume. If you want to try it in your own environment, do so at your own risk.

Stacking Field Definition

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT_IP / IP</td>
<td>IP address running queries</td>
</tr>
<tr>
<td>USERNAME</td>
<td>Name of the user account that ran the query</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>The name of the application that ran queries, from CLIENT_ENVIRONMENT</td>
</tr>
<tr>
<td>EARLIEST.UTC</td>
<td>First time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>LATEST.UTC</td>
<td>Last time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>UNIQUE QUERIES</td>
<td>Count of unique queries per grouped variation</td>
</tr>
<tr>
<td>TOTAL QUERIES</td>
<td>Count of total queries per grouped variation</td>
</tr>
<tr>
<td>DAY COUNT</td>
<td>Count of unique days active per grouped variation</td>
</tr>
<tr>
<td>UNIQ_USER</td>
<td>Count of unique users per grouped variation</td>
</tr>
<tr>
<td>ARR_UNIQ_USER</td>
<td>Array of users that ran queries per grouped variation</td>
</tr>
<tr>
<td>UNIQ_APP_NAME</td>
<td>Count of unique application names per grouped variation</td>
</tr>
<tr>
<td>ARR_APP_NAME</td>
<td>Array of application names per grouped variation</td>
</tr>
<tr>
<td>UNIQ_OS</td>
<td>Count of unique operating systems per grouped variation</td>
</tr>
<tr>
<td>ARR_OP_SYS</td>
<td>Array of operating systems per grouped variation</td>
</tr>
<tr>
<td>UNIQ_IP</td>
<td>Count of unique IPs per grouped variation</td>
</tr>
</tbody>
</table>

Query 1: Spikes in View Access by User, Application Name, IP, and Operating System Across Whole Snowflake Environment Per Day

Query

```sql
WITH sq AS (   
  Unset
)   
```
SELECT START_TIME, 
    DATE(START_TIME) AS QUERY_DATE, 
    QUERY_HASH, 
    QUERY_TEXT, 
    SESSION_ID, 
    REGEXP_REPLACE(REGEXP_SUBSTR(QUERY_TEXT, $$FROM.*\[(\w+).\[(\w+)].\[(\w+)\]$$,1,1,'mse'),""','"') as DB_S_TV_STR,
CASE
    WHEN UPPER(QUERY_TEXT) LIKE '%'COUNT('%' THEN 'TRUE'
    ELSE 'FALSE'
END as IS_COUNT_Q
FROM snowflake.account_usage.query_history q
WHERE EXECUTION_STATUS = 'SUCCESS' AND
    (upper(QUERY_TEXT) LIKE 'SELECT%' OR upper(QUERY_TEXT) LIKE 'COPY INTO%') AND
    upper(QUERY_TEXT) LIKE '%FROM%' AND
    NOT REGEXP_LIKE(UPPER(QUERY_TEXT),'.*'INFORMATION_SCHEMA.*','s')
)

SELECT QUERY_DATE,
    COUNT(DISTINCT sq.QUERY_HASH) AS UNIQ_QUERIES,
    COUNT(DISTINCT l.USER_NAME) AS UNIQ_USER,
    COUNT(DISTINCT l.CLIENT_IP) AS UNIQ_IP,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APPS,
    COUNT(DISTINCT sq.SESSION_ID) AS UNIQ_SESSION,
    SPLIT_PART(DB_S_TV_STR, '.', 1) AS target_DB,
    SPLIT_PART(DB_S_TV_STR, '.', 2) AS target_schema,
    SPLIT_PART(DB_S_TV_STR, '.', 3) AS target_table,
FROM sq
JOIN snowflake.account_usage.sessions s ON s.session_id = sq.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = s.LOGIN_EVENT_ID
WHERE sq.IS_COUNT_Q = 'FALSE' AND
    DB_S_TV_STR IS NOT NULL
GROUP BY
    QUERY_DATE,
    target_DB,
    target_schema,
    target_table
Query 2: Stacking Table and DB Access Application Name, IP, and Operating System per Unique User and Day

Query

```sql
WITH sq AS (  
SELECT  
  START_TIME,  
  DATE(START_TIME) AS QUERY_DATE,  
  QUERY_HASH,  
  QUERY_TEXT,  
  SESSION_ID,  
  REGEXP_REPLACE(REGEXP_SUBSTR(QUERY_TEXT,  
  $$FROM.*\W\([\w]+.\[\w]+.\[\w]+\).*$$,1,1,'mse'),'"','') as DB_S_TV_STR,  
  CASE  
    WHEN UPPER(QUERY_TEXT) LIKE '%COUNT%' THEN 'TRUE'  
  ELSE 'FALSE'  
  END as IS_COUNT_Q  
FROM snowflake.account_usage.query_history q  
WHERE  
  EXECUTION_STATUS = 'SUCCESS' AND  
  (upper(QUERY_TEXT) LIKE 'SELECT%' OR upper(QUERY_TEXT) LIKE 'COPY INTO%') AND  
  upper(QUERY_TEXT) LIKE '%FROM%' AND  
  NOT REGEXP_LIKE(UPPER(QUERY_TEXT),'.*INFORMATION_SCHEMA.*','s')  
)

SELECT  
  QUERY_DATE,  
  l.USER_NAME AS USERNAME,  
  COUNT(DISTINCT sq.QUERY_HASH) AS UNIQ_QUERIES,  
  SPLIT_PART(DB_S_TV_STR, '.', 1) AS target_DB,  
  SPLIT_PART(DB_S_TV_STR, '.', 2) AS target_schema,  
  SPLIT_PART(DB_S_TV_STR, '.', 3) AS target_table,  
  COUNT(DISTINCT l.CLIENT_IP) AS UNIQ_IP,  
  COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APPS,  
  SPLIT_PART(DB_S_TV_STR, '.', 1) AS target_DB,  
  SPLIT_PART(DB_S_TV_STR, '.', 2) AS target_schema,  
  SPLIT_PART(DB_S_TV_STR, '.', 3) AS target_table,  
  COUNT(DISTINCT sq.SESSION_ID) AS UNIQ_SESSION  
FROM sq, snowflake.account_usage.query_history q, snowflake.account_usage.query_history l  
WHERE  
  l.SESSION_ID = sq.SESSION_ID  
GROUP BY  
  QUERY_DATE,  
  l.USER_NAME,  
  DB_S_TV_STR,  
  APPLICATION,  
  UNIQ_QUERIES,  
  UNIQ_IP,  
  UNIQ_APPS,  
  UNIQ_SESSION  
);  

© Copyright 2024 Mandiant
FROM sq
  JOIN snowflake.account_usage.sessions s ON s.session_id = sq.session_id
  JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = s.LOGIN_EVENT_ID
WHERE
  sq.IS_COUNT_Q = 'FALSE' AND
  DB_S_TV_STR IS NOT NULL
GROUP BY
  QUERY_DATE,
  USERNAME,
  target_DB,
  target_schema,
  target_table

Query 3: Stacking Table and DB Access per User, IP, and Operating System per Unique Application Used and Day

Query

WITH sq AS (  
SELECT
  START_TIME,
  DATE(START_TIME) AS QUERY_DATE,
  QUERY_HASH,
  QUERY_TEXT,
  SESSION_ID,
  REGEXP_REPLACE(REGEXP_SUBSTR(QUERY_TEXT, $$FROM.*\W\([\w]+.\[\w]+.\[\w]+\).*$$,1,1,'mse'),"","\"\"",\'\'\',\'\') as DB_S_TV_STR,
  CASE
    WHEN UPPER(QUERY_TEXT) LIKE '%COUNT%' THEN 'TRUE'
    ELSE 'FALSE'
  END as IS_COUNT_Q
FROM snowflake.account_usage.query_history q
WHERE
  EXECUTION_STATUS = 'SUCCESS' AND
  (upper(QUERY_TEXT) LIKE 'SELECT%' OR upper(QUERY_TEXT) LIKE 'COPY INTO%') AND
  upper(QUERY_TEXT) LIKE '%FROM%' AND
  NOT REGEXP_LIKE(UPPER(QUERY_TEXT),'.*INFORMATION_SCHEMA.*','s')
)
SELECT
    QUERY_DATE,
    JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION') AS APPLICATION_NAME,
    COUNT(DISTINCT sq.QUERY_HASH) AS UNIQ_QUERIES,
    COUNT(DISTINCT l.USER_NAME) AS UNIQ_USER,
    COUNT(DISTINCT l.CLIENT_IP) AS UNIQ_IP,
    COUNT(DISTINCT sq.SESSION_ID) AS UNIQ_SESSION,
    SPLIT_PART(DB_S_TV_STR, '.', 1) AS target_DB,
    SPLIT_PART(DB_S_TV_STR, '.', 2) AS target_schema,
    SPLIT_PART(DB_S_TV_STR, '.', 3) AS target_table,
FROM sq
    JOIN snowflake.account_usage.sessions s ON s.session_id = sq.session_id
    JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = s.LOGIN EVENT_ID
WHERE
    sq.IS_COUNT_Q = 'FALSE' AND
    DB_S_TV_STR IS NOT NULL
GROUP BY
    QUERY_DATE,
    APPLICATION_NAME,
    target_DB,
    target_schema,
    target_table

User Creation and Deletion

Description

The “Snowflake.Account_Usage.Users” view can be utilized to identify all users currently and historically present the target Snowflake account over the past year. In addition to creation, deletion, and last password modification timestamps, other properties surrounding the account such as role, MFA enforcement, and email address are also visible.

Potential Attacker Trends

Depending on the end goal of the attacker, behavior will vary per attack lifecycle. Focusing on anomalies such as

- Account creations (CREATED_ON) followed by rapid deletions (DELETED_ON)
- Invalid or abnormal domains present in email addresses (EMAIL)
- Abnormal password reset times (PASSWORD_LAST_SET_TIME)
- Disabled MFA (EXT_AUTHN_DUO) in an environment where it is normally required

Field Definition
The field definition for the users view can be found in the following Snowflake documentation:

Query 1: Full export of Users View

Query

```
SELECT * FROM snowflake.account_usage.users
```
Query Analysis

Frequency Analysis

Description
When performing recon on the environment, the number of queries performed by the attacker were observed as a spike comparatively to those seen across normal days within the environment. The queries below break out different ways of stacking the number of queries based on User, Application Name, IP, and Operating System.

Potential Attacker Trends
Look for abnormal spikes across user accounts and IP addresses (specifically for service accounts with relatively fixed rates of queries day per day due to automations).

Stacking Field Definition

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT_IP / IP</td>
<td>IP address running queries</td>
</tr>
<tr>
<td>USERNAME</td>
<td>Name of the user account that ran the query</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>The name of the application that ran queries, from CLIENT_ENVIRONMENT</td>
</tr>
<tr>
<td>EARLIEST_UTC</td>
<td>First time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>LATEST_UTC</td>
<td>Last time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>UNIQUE_QUERIES</td>
<td>Count of unique queries per grouped variation</td>
</tr>
<tr>
<td>TOTAL_QUERIES</td>
<td>Count of total queries per grouped variation</td>
</tr>
<tr>
<td>DAY_COUNT</td>
<td>Count of unique days active per grouped variation</td>
</tr>
<tr>
<td>UNIQ_USER</td>
<td>Count of unique users per grouped variation</td>
</tr>
<tr>
<td>ARR_UNIQ_USER</td>
<td>Array of users that ran queries per grouped variation</td>
</tr>
<tr>
<td>UNIQ_APP_NAME</td>
<td>Count of unique application names per grouped variation</td>
</tr>
<tr>
<td>ARR_APP_NAME</td>
<td>Array of application names per grouped variation</td>
</tr>
<tr>
<td>UNIQ_OS</td>
<td>Count of unique operating systems per grouped variation</td>
</tr>
</tbody>
</table>

© Copyright 2024 Mandiant
<table>
<thead>
<tr>
<th>ARR_OP_SYS</th>
<th>Array of operating systems per grouped variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIQ_IP</td>
<td>Count of unique IPs per grouped variation</td>
</tr>
</tbody>
</table>

Query 1: Spikes in Query Count per Day (Showing Distinct and Count of Users, Application Names, Operating System, and IP)

Query

```sql
SELECT
    DATE(START_TIME) AS QUERY_DATE,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS UNIQ_OS,
    ARRAY_UNIQUE_AGG(l.CLIENT_IP) AS ARR_IP,
    COUNT(DISTINCT l.CLIENT_IP) AS UNIQ_IP,
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history ON l.EVENT_ID = s.LOGIN_EVENT_ID
GROUP BY
    QUERY_DATE
```
Query 2: Query Frequency & Behavior per User

Query

```
SELECT
    q.USER_NAME AS USERNAME,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT,'APPLICATION')) AS ARR_APP_NAME,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT,'APPLICATION')) AS UNIQ_APP_NAME,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT,'OS')) AS ARR_OP_SYS,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT,'OS')) AS UNIQ_OS,
    ARRAY_UNIQUE_AGG(l.CLIENT_IP) AS ARR_IP,
    COUNT(DISTINCT l.CLIENT_IP) AS UNIQ_IP
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history ON l.EVENT_ID = s.LOGIN_EVENT_ID
GROUP BY
    USERNAME
```

Query 3: Query Frequency & Behavior per IP

Query

```
SELECT
    l.CLIENT_IP AS IP,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
```
Query 4: Query Frequency & Behavior per Application Used

Query

```sql
TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)), 'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS,
COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS UNIQ_OS,
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history ON l.EVENT_ID = s.LOGIN_EVENT_ID
GROUP BY q
```
Error Rate Analysis

Description

Similar to query frequency analysis, attackers often have abnormal failure/success rates as compared to service and user account norms. Attacker brute force tools and fuzzers generate failures while enumerating the DB schemas, while on the flipside a misconfigured service account dropping in error % per day for a short span of time without explanation can be indicative of interactive user activity on the account.

Observations

After identifying the error codes of interest (the top 4 with deviations during the time of interest), failed transactions were filtered by user, date, and code. These can then be post-processed to timechart the errors by user, and filtered by codes.

Warnings About FP/Noise

Error codes are 6 digit, zero padded, but the padded zeros are not visible in the output results. Padded zeros must be explicitly specified if using IN/NOT IN conditions within the WHERE statement.

An initial stack on error codes by DATE only will help identify the noisiest codes, which can then be filtered out.

Since the user count can be very high, it is worth running different filters to check named-user and non-named-user separately. In our case, named users had the format F.L, so NOT USERNAME LIKE '%.%' will remove named users.

Stacking Field Definition

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT_IP / IP</td>
<td>IP address running queries</td>
</tr>
<tr>
<td><strong>USERNAME</strong></td>
<td>Name of the user account that ran the query</td>
</tr>
<tr>
<td><strong>APPLICATION_NAME</strong></td>
<td>The name of the application that ran queries, from CLIENT_ENVIRONMENT</td>
</tr>
<tr>
<td><strong>EARLIEST_UTC</strong></td>
<td>First time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td><strong>LATEST_UTC</strong></td>
<td>Last time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td><strong>UNIQUE_QUERIES</strong></td>
<td>Count of unique queries per grouped variation</td>
</tr>
<tr>
<td><strong>TOTAL_QUERIES</strong></td>
<td>Count of total queries per grouped variation</td>
</tr>
<tr>
<td><strong>DAY_COUNT</strong></td>
<td>Count of unique days active per grouped variation</td>
</tr>
<tr>
<td><strong>UNIQ_USER</strong></td>
<td>Count of unique users per grouped variation</td>
</tr>
<tr>
<td><strong>ARR_UNIQ_USER</strong></td>
<td>Array of users that ran queries per grouped variation</td>
</tr>
<tr>
<td><strong>UNIQ_APP_NAME</strong></td>
<td>Count of unique application names per grouped variation</td>
</tr>
<tr>
<td><strong>ARR_APP_NAME</strong></td>
<td>Array of application names per grouped variation</td>
</tr>
<tr>
<td><strong>UNIQ_OS</strong></td>
<td>Count of unique operating systems per grouped variation</td>
</tr>
<tr>
<td><strong>ARR_OP_SYS</strong></td>
<td>Array of operating systems per grouped variation</td>
</tr>
<tr>
<td><strong>UNIQ_IP</strong></td>
<td>Count of unique IPs per grouped variation</td>
</tr>
<tr>
<td><strong>UNIQ_CODES</strong></td>
<td>Count of unique error codes</td>
</tr>
<tr>
<td><strong>UNIQ_MSG</strong></td>
<td>Count of unique error messages</td>
</tr>
<tr>
<td><strong>UNIQ_CODE_MSG_RATE</strong></td>
<td>UNIQ_CODES divided by the UNIQ_MSG</td>
</tr>
<tr>
<td><strong>UNIQ_MSG_TOTAL_RATE</strong></td>
<td>UNIQ_MSG divided by the ERR_COUNT</td>
</tr>
<tr>
<td><strong>Q_ERR_CODE</strong></td>
<td>Query error code</td>
</tr>
<tr>
<td><strong>FAIL_COUNT</strong></td>
<td>Count of total &quot;fail&quot; results for queries run</td>
</tr>
<tr>
<td><strong>SUCCESS_COUNT</strong></td>
<td>Count of total &quot;success&quot; results for queries run</td>
</tr>
<tr>
<td><strong>FAIL_DAY_COUNT</strong></td>
<td>Count of unique days on which a fail occurred</td>
</tr>
<tr>
<td><strong>AVG_QUERY_PER_DAY</strong></td>
<td>Average queries run per day</td>
</tr>
</tbody>
</table>
Query 1: Identifying Which Queries Result in the Highest Error Rate

Query

```
SELECT
    DATE(START_TIME) AS QUERY_DATE,
    COUNT( DISTINCT CLIENT_IP) AS UNIQ_IP,
    COUNT(DISTINCT q.ERROR_CODE) AS UNIQ_CODES,
    COUNT(DISTINCT q.ERROR_MESSAGE) AS UNIQ_MSG,
    COUNT(q.ERROR_MESSAGE) AS ERR_COUNT,
    TO_NUMBER((UNIQ_CODES/UNIQ_MSG)*100, 10, 2) AS UNIQ_CODE_MSG_RATE,
    TO_NUMBER((UNIQ_MSG/ERR_COUNT)*100, 10, 2) AS UNIQ_MSG_TOTAL_RATE
FROM snowflake.account_usage.query_history q
    JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
    JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
WHERE
    EXECUTION_STATUS = 'FAIL' AND NOT q.ERROR_CODE IN ('090106', '002129')
GROUP BY
    QUERY_DATE
ORDER BY
    QUERY_DATE
```

Query 2: Frequency of Errors by Error Reason Type (Codes, etc.)

Query

```
SELECT
    q.ERROR_CODE AS Q_ERR_CODE,
    COUNT(DISTINCT q.ERROR_MESSAGE) AS UNIQ_ERR_MSG,
    COUNT(EXECUTION_STATUS) AS HIT_COUNT,
    COUNT( DISTINCT CLIENT_IP) AS UNIQ_IP,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS
FROM snowflake.account_usage.query_history q
    JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
    JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
```
WHERE
   EXECUTION_STATUS = 'FAIL'
GROUP BY
   Q_ERR_CODE
ORDER BY
   Q_ERR_CODE

Query 3: Frequency of Errors per IP

Query

WITH
   FAIL_DATES AS (
      SELECT
         1.CLIENT_IP AS IP,
         TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_FAILUtc,
         TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS LATEST_FAILUtc,
         COUNT(DISTINCT DATE(START_TIME)) AS FAIL_DAY_COUNT,
      FROM
         snowflake.account_usage.query_history
      JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
      JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
      WHERE EXECUTION_STATUS = 'FAIL'
      GROUP BY IP
   )

SELECT * FROM
   (SELECT
      CLIENT_IP,
      COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
      SUM(FAIL_DAY_COUNT) AS FAIL_DAY_COUNT,
      COUNT_IF(EXECUTION_STATUS = 'FAIL') AS FAIL_COUNT,
      COUNT_IF(EXECUTION_STATUS = 'SUCCESS') AS SUCCESS_COUNT,
      COUNT(EXECUTION_STATUS) AS HIT_COUNT,
      TO_NUMBER((FAIL_COUNT/HIT_COUNT)*100, 10, 2) AS FAIL_RATE,
TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
MIN(EARLIEST_FAIL_UTC) AS EARLIEST_FAIL_UTC,
MAX(LATEST_FAIL_UTC) AS LATEST_FAIL_UTC,
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
JOIN FAIL_DATES ON FAIL_DATES.IP = l.CLIENT_IP
GROUP BY
    CLIENT_IP
)
ORDER BY
    CLIENT_IP

Query 4: Frequency of Errors per User

Query

WITH FAIL_DATES AS (  
SELECT  
    l.USER_NAME AS USERNAME,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
    COUNT( DISTINCT DATE(START_TIME) ) AS FAIL_DAY_COUNT,
FROM
    snowflake.account_usage.query_history
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
    WHERE EXECUTION_STATUS = 'FAIL'
GROUP BY USERNAME
)
SELECT * FROM
  (SELECT
      USERNAME,
      COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
      SUM(FAIL_DAY_COUNT) AS FAIL_DAY_COUNT,
      COUNT_IF(EXECUTION_STATUS = 'FAIL') AS FAIL_COUNT,
      COUNT_IF(EXECUTION_STATUS = 'SUCCESS') AS SUCCESS_COUNT,
      COUNT(EXECUTION_STATUS) AS HIT_COUNT,
      TO_NUMBER((FAIL_COUNT/HIT_COUNT)*100, 10, 2) AS FAIL_RATE,
      TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
      TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
      MIN(EARLIEST_FAIL_UTC) AS EARLIEST_FAIL_UTC,
      MAX(LATEST_FAIL_UTC) AS LATEST_FAIL_UTC
  FROM snowflake.account_usage.query_history q
  JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
  JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
  JOIN FAIL_DATES ON FAIL_DATES.USERNAME = l.USER_NAME
  GROUP BY
      USERNAME
  )
ORDER BY
      USERNAME

Query 5: Frequency of Errors per Application Name

Query

WITH
    FAIL_DATES AS (
      SELECT
         JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION') AS APPLICATION_NAME,
         s.CLIENT_ENVIRONMENT AS JSON_BLOB,
         s.CLIENT_ENVIRONMENT AS JSON_BLOB,
    )
TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_FAIL_UTC,
TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS LATEST_FAIL_UTC,
COUNT(DISTINCT DATE(START_TIME)) AS FAIL_DAY_COUNT,
FROM
snowflake.account_usage.query_history
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
WHERE EXECUTION_STATUS = 'FAIL'
GROUP BY APPLICATION_NAME, JSON_BLOB
)
SELECT
APPLICATION_NAME,
COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
SUM(FAIL_DAY_COUNT) AS FAIL_DAY_COUNT,
COUNT_IF(EXECUTION_STATUS = 'FAIL') AS FAIL_COUNT,
COUNT_IF(EXECUTION_STATUS = 'SUCCESS') AS SUCCESS_COUNT,
COUNT(EXECUTION_STATUS) AS HIT_COUNT,
TO_NUMBER((FAIL_COUNT/HIT_COUNT)*100, 10, 2) AS FAIL_RATE,
TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
MIN(EARLIEST_FAIL_UTC) AS EARLIEST_FAIL_UTC,
MAX(LATEST_FAIL_UTC) AS LATEST_FAIL_UTC
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
JOIN FAIL_DATES ON FAIL_DATES.JSON_BLOB = s.CLIENT_ENVIRONMENT
GROUP BY
APPLICATION_NAME
ORDER BY
APPLICATION_NAME
Query 6: Error Count, Code by User, Date, Select Code Inclusions

Query

```
SELECT
    l.USER_NAME AS USERNAME,
    DATE(START_TIME) AS QUERY_DATE,
    q.ERROR_CODE AS Q_ERR_CODE,
    COUNT(DISTINCT CLIENT_IP) AS UNIQ_IP,
    COUNT(DISTINCT q.ERROR_CODE) AS UNIQ_CODES,
    COUNT(DISTINCT q.ERROR_MESSAGE) AS UNIQ_MSG,
    COUNT(q.ERROR_MESSAGE) AS ERR_COUNT,
    TO_NUMBER((UNIQ_CODES/UNIQ_MSG)*100, 10, 2) AS UNIQ_CODE_MSG_RATE,
    TO_NUMBER((UNIQ_MSG/ERR_COUNT)*100, 10, 2) AS UNIQ_MSG_TOTAL_RATE
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
WHERE
    EXECUTION_STATUS = 'FAIL' AND q.ERROR_CODE IN ('<Code 1>', '<Code 2>', '<Code 3>', '<Code 4>')</noset>
GROUP BY
    QUERY_DATE,
    USERNAME,
    Q_ERR_CODE
```

High Resource Consumption

Description

Analysis of resources and cloud credits used for queries. When attackers aim to exfiltrate large amounts of data, the resources used to perform the queries/compression/exfil increase the compute resources required above the norm.

Warnings About FP/Noise

Metrics for CREDITS_USED_CLOUD_SERVICES are often directly related to the computational power needed to run a query; however, computational power is not always indicative of attacker queries. Since an attacker is often enumerating and exfiltrating data, ROWS_PRODUCED and ROWS_WRITTEN_TO_RESULT fields are more likely to help identify the latter.

© Copyright 2024 Mandiant
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT_IP / IP</td>
<td>IP address running queries</td>
</tr>
<tr>
<td>USERNAME</td>
<td>Name of the user account that ran the query</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>The name of the application that ran queries, from CLIENT_ENVIRONMENT</td>
</tr>
<tr>
<td>EARLIEST_UTC</td>
<td>First time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>LATEST_UTC</td>
<td>Last time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>UNIQUE_QUERIES</td>
<td>Count of unique queries per grouped variation</td>
</tr>
<tr>
<td>TOTAL_QUERIES</td>
<td>Count of total queries per grouped variation</td>
</tr>
<tr>
<td>DAY_COUNT</td>
<td>Count of unique days active per grouped variation</td>
</tr>
<tr>
<td>UNIQ_USER</td>
<td>Count of unique users per grouped variation</td>
</tr>
<tr>
<td>ARR_UNIQ_USER</td>
<td>Array of users that ran queries per grouped variation</td>
</tr>
<tr>
<td>UNIQ_APP_NAME</td>
<td>Count of unique application names per grouped variation</td>
</tr>
<tr>
<td>ARR_APP_NAME</td>
<td>Array of application names per grouped variation</td>
</tr>
<tr>
<td>UNIQ_OS</td>
<td>Count of unique operating systems per grouped variation</td>
</tr>
<tr>
<td>ARR_OP_SYS</td>
<td>Array of operating systems per grouped variation</td>
</tr>
<tr>
<td>UNIQ_IP</td>
<td>Count of unique IPs per grouped variation</td>
</tr>
<tr>
<td>QUERY_LOAD_PERCENT</td>
<td>Summation of the query load percent per grouped variation</td>
</tr>
<tr>
<td>QUERY_ACCELERATIONBYTES_SCANNED</td>
<td>Summation of the query acceleration bytes scanned per grouped variation</td>
</tr>
<tr>
<td>QUERY_ACCELERATION_PARTITIONS_SCANNED</td>
<td>Summation of the query acceleration partitions scanned per grouped variation</td>
</tr>
<tr>
<td>QUERY_ACCELERATION_UPPER_LIMIT_SCALE_FACTOR</td>
<td>Summation of the query acceleration scale factor per grouped variation</td>
</tr>
<tr>
<td>ARR_OUTBOUND_DATATRANSFER_CLOUD</td>
<td>Array of unique values for outbound data transfer cloud (AWS, AZURE etc.)</td>
</tr>
<tr>
<td><strong>ARR_OUTBOUND_DATA_TRANSFER_REGION</strong></td>
<td>Array of unique values for outbound cloud regions (east, west, etc.)</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>OUTBOUND_DATA_TRANSFER_BYTES</strong></td>
<td>Summation of the outbound data transfer bytes</td>
</tr>
<tr>
<td><strong>ARR_INBOUND_DATA_TRANSFER_CLOUD</strong></td>
<td>Array of unique values for inbound data transfer cloud (AWS, AZURE etc.)</td>
</tr>
<tr>
<td><strong>ARR_INBOUND_DATA_TRANSFER_REGION</strong></td>
<td>Array of unique values for inbound cloud regions (east, west, etc.)</td>
</tr>
<tr>
<td><strong>INBOUND_DATA_TRANSFER_BYTES</strong></td>
<td>Summation of the inbound data transfer bytes</td>
</tr>
<tr>
<td><strong>CREDITS_USED_CLOUD_SERVICES</strong></td>
<td>Summation of the cloud credits used</td>
</tr>
<tr>
<td><strong>BYTES_WRITTEN</strong></td>
<td>Summation of the bytes written</td>
</tr>
<tr>
<td><strong>BYTES_WRITTEN_TO_RESULT</strong></td>
<td>Summation of the bytes written to result</td>
</tr>
<tr>
<td><strong>BYTES_READ_FROM_RESULT</strong></td>
<td>Summation of the bytes read from result</td>
</tr>
<tr>
<td><strong>ROWS_PRODUCED</strong></td>
<td>Summation of the rows produced</td>
</tr>
<tr>
<td><strong>ROWS_INSERTED</strong></td>
<td>Summation of the rows inserted</td>
</tr>
<tr>
<td><strong>ROWS_UPDATED</strong></td>
<td>Summation of the rows updated</td>
</tr>
<tr>
<td><strong>ROWS_DELETED</strong></td>
<td>Summation of the rows deleted</td>
</tr>
<tr>
<td><strong>ROWS_UNLOADED</strong></td>
<td>Summation of the rows unloaded</td>
</tr>
<tr>
<td><strong>ROWS_WRITTEN_TO_RESULT</strong></td>
<td>Summation of the rows written to result</td>
</tr>
</tbody>
</table>
Query 1: Identifying Count of Queries with High Resource Consumption

Query

```sql
UNSET
SELECT
    DATE(START_TIME) AS QUERY_DATE,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
    SUM(QUERY_LOAD_PERCENT) AS QUERY_LOAD_PERCENT,
    SUM(QUERY_ACCELERATION_BYTES_SCANNED) AS QUERY_ACCELERATION_BYTES_SCANNED,
    SUM(QUERY_ACCELERATION_PARTITIONS_SCANNED) AS QUERY_ACCELERATION_PARTITIONS_SCANNED,
    SUM(QUERY_ACCELERATION_UPPER_LIMIT_SCALE_FACTOR) AS QUERY_ACCELERATION_UPPER_LIMIT_SCALE_FACTOR,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = s.LOGIN_EVENT_ID
GROUP BY QUERY_DATE
```

Query 2: Identifying High Credit Usage

Query

```sql
UNSET
SELECT
    DATE(START_TIME) AS QUERY_DATE,
    JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION') AS APPLICATION_NAME,
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = s.LOGIN_EVENT_ID
```
SELECT DATE(START_TIME) AS QUERY_DATE,
        1.CLIENT_IP AS IP,
        SUM(BYTES_WRITTEN) AS BYTES_WRITTEN,
        SUM(BYTES_WRITTEN_TO_RESULT) AS BYTES_WRITTEN_TO_RESULT,
Query 4: Identify Resource Consumption per User

Query

```sql
SELECT
    DATE(START_TIME) AS QUERY_DATE,
    l.USER_NAME AS USERNAME,
    SUM(BYTES_WRITTEN) AS BYTES_WRITTEN,
    SUM(BYTES_WRITTEN_TO_RESULT) AS BYTES_WRITTEN_TO_RESULT,
    SUM(BYTES_READ_FROM_RESULT) AS BYTES_READ_FROM_RESULT,
    SUM(ROWS_PRODUCED) AS ROWS_PRODUCED,
    SUM(ROWS_INSERTED) AS ROWS_INSERTED,
    SUM(ROWS_UPDATED) AS ROWS_UPDATED,
    SUM(ROWS_DELETED) AS ROWS_DELETED,
    SUM(ROWS_UNLOADED) AS ROWS_UNLOADED,
    SUM(ROWS_WRITTEN_TO_RESULT) AS ROWS_WRITTEN_TO_RESULT,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)) ,'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)) ,'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = s.LOGIN_EVENT_ID
GROUP BY
    QUERY_DATE,
    IP
```
SUM(ROWS_INSERTED) AS ROWS_INSERTED,
SUM(ROWS_UPDATED) AS ROWS_UPDATED,
SUM(ROWS_DELETED) AS ROWS_DELETED,
SUM(ROWS_UNLOADED) AS ROWS_UNLOADED,
SUM(ROWS_WRITTEN_TO_RESULT) AS ROWS_WRITTEN_TO_RESULT,
COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = s.LOGIN_EVENT_ID
GROUP BY
QUERY_DATE,
USERNAME

Query 5: Stack by App

Query

SELECT
    DATE(START_TIME) AS QUERY_DATE,
    JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION') AS APPLICATION_NAME,
    SUM(BYTES_WRITTEN) AS BYTES_WRITTEN,
    SUM(BYTES_WRITTEN_TO_RESULT) AS BYTES_WRITTEN_TO_RESULT,
    SUM(BYTES_READ_FROM_RESULT) AS BYTES_READ_FROM_RESULT,
    SUM(ROWS_PRODUCED) AS ROWS_PRODUCED,
    SUM(ROWS_INSERTED) AS ROWS_INSERTED,
    SUM(ROWS_UPDATED) AS ROWS_UPDATED,
Long Running Queries

Description
The following queries were designed to identify queries with abnormally long run times compared to the average run within the snowflake environment. Two separate queries were designed to break out Long Running Query Usage - Large amounts of abnormally long queries from any single user or IP address.

Potential Attacker Trends
Please see the "Indicators of Compromise (IOCs)" section of the blog post for an actively updated list of Applications observed by UNC5537.

All Observed Variations (If Relevant)

- Snowflake logging/app parsing is not bulletproof. In some cases, the application name "C:\\Program" is seen in the s.CLIENT_ENVIRONMENT JSON blob, potentially caused by
the space in "Program Files." We have seen situations where the Windows Server 2022 compromised account and bad IP use this application name and believe it might be truncating the DBeaver app name during logging.

- For reference, the functions PARSE_JSON() and JSON_EXTRACT_PATH_TEXT can be used to extract/transform these fields.

Warnings About FP/Noise

- Although SnowSQL was observed linked to TA activity, we saw historic use of SnowSQL for additional uncompromised accounts back in early 2023. This app name alone is not a high-fidelity indicator, but should be paired with other findings.
- com.amazonaws.services.glue.ProcessLauncher and PythonConnector were our two noisiest app names, and filtering these out reduced our dataset from 60k down to around 500.

Stacking Field Definition

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT_IP / IP</td>
<td>IP address running queries</td>
</tr>
<tr>
<td>USERNAME</td>
<td>Name of the user account that ran the query</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>The name of the application that ran queries, from CLIENT_ENVIRONMENT</td>
</tr>
<tr>
<td>EARLIEST_UTC</td>
<td>First time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>LATEST_UTC</td>
<td>Last time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>UNIQUE_QUERIES</td>
<td>Count of unique queries per grouped variation</td>
</tr>
<tr>
<td>TOTAL_QUERIES</td>
<td>Count of total queries per grouped variation</td>
</tr>
<tr>
<td>DAY_COUNT</td>
<td>Count of unique days active per grouped variation</td>
</tr>
<tr>
<td>UNIQ_USER</td>
<td>Count of unique users per grouped variation</td>
</tr>
<tr>
<td>ARR_Uniq_USER</td>
<td>Array of users that ran queries per grouped variation</td>
</tr>
<tr>
<td>UNIQ_APP_NAME</td>
<td>Count of unique application names per grouped variation</td>
</tr>
<tr>
<td>ARR_APP_NAME</td>
<td>Array of application names per grouped variation</td>
</tr>
<tr>
<td>UNIQ_OS</td>
<td>Count of unique operating systems per grouped variation</td>
</tr>
<tr>
<td>ARR_OP_SYS</td>
<td>Array of operating systems per grouped variation</td>
</tr>
</tbody>
</table>
Query 1: Stack by IP

SELECT
    CLIENT_IP,
    COUNT(DISTINCT QUERY_TEXT) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT,'APPLICATION')) AS ARR_APP_NAME,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT,'APPLICATION')) AS UNIQ_APP_NAME,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT,'OS')) AS ARR_OP_SYS,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT,'OS')) AS UNIQ_OS,
    ARRAY_UNIQUE_AGG(l.USER_NAME) AS ARR_USER,
    COUNT(DISTINCT l.USER_NAME) AS UNIQ_USER
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID =
    q.LOGIN_EVENT_ID
WHERE
    q.total_elapsed_time >
    (SELECT AVG(TOTAL_ELAPSED_TIME) FROM snowflake.account_usage.query_history q )
GROUP BY CLIENT_IP
Query 2: Stack by User

Query

```sql
SELECT
    q.USER_NAME,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS UNIQ_OS,
    ARRAY_UNIQUE_AGG(l.CLIENT_IP) AS ARR_IP,
    COUNT(DISTINCT l.CLIENT_IP) AS UNIQ_IP,
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
WHERE q.total_elapsed_time >
    (SELECT AVG(TOTAL_ELAPSED_TIME) FROM snowflake.account_usage.query_history q)
GROUP BY q.USER_NAME
```

Multi-Day Duplicate Queries

Description

Several engagements have shown the attacker running many of the same queries across a small selection of days. The below query can be run with your known bad IPs to identify if this multi-day repeat pattern of queries exists in a client environment.
Potential Attacker Trends

We observed the attacker running several thousand duplicate queries across three distinct days. These will clearly stick out from the rest of the attacker’s activity.

Warnings About FP/Noise

Caution should be taken not to stack this query by username. Since the attacker may be using an existing account, which itself may be used to run recurring queries, it will have a tendency to show a high amount of noise.

Query 1: Multi-day repeats

Query

```
WITH sq AS (
  SELECT
    QUERY_HASH,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT
  FROM snowflake.account_usage.query_history q
  JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
  JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = s.LOGIN_EVENT_ID
  WHERE l.CLIENT_IP IN ('IP-1', 'IP-2', ... 'IP-n')
  GROUP BY QUERY_HASH
)
SELECT // pull queries from 1, dates from 2, stack by 1) date 2) raw query text
  DATE(START_TIME) AS QUERY_DATE,
  COUNT(DISTINCT q.QUERY_HASH) AS UNIQUE_QUERIES,
  ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
  ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS,
  ARRAY_UNIQUE_AGG(l.CLIENT_IP) AS ARR_IP,
  ARRAY_UNIQUE_AGG(l.USER_NAME) AS ARR_USERNAME
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = s.LOGIN_EVENT_ID
JOIN sq ON sq.QUERY_HASH = q.QUERY_HASH
WHERE
```
sq.DAY_COUNT > 1
AND
l.CLIENT_IP IN ('IP-1', 'IP-2', ... 'IP-n')
GROUP BY
  QUERY_DATE
ORDER BY
  QUERY_DATE ASC
Staging/Exfil

Data Compression

Description
Analysis of queries that use commands associated with compression, often seen prior to exfil.

Stacking Field Definition

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT_IP / IP</td>
<td>IP address running queries</td>
</tr>
<tr>
<td>USERNAME</td>
<td>Name of the user account that ran the query</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>The name of the application that ran queries, from CLIENT_ENVIRONMENT</td>
</tr>
<tr>
<td>EARLIEST_UTC</td>
<td>First time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>LATEST_UTC</td>
<td>Last time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>UNIQUE_QUERIES</td>
<td>Count of unique queries per grouped variation</td>
</tr>
<tr>
<td>TOTAL_QUERIES</td>
<td>Count of total queries per grouped variation</td>
</tr>
<tr>
<td>DAY_COUNT</td>
<td>Count of unique days active per grouped variation</td>
</tr>
<tr>
<td>UNIQ_USER</td>
<td>Count of unique users per grouped variation</td>
</tr>
<tr>
<td>ARR_UNIQ_USER</td>
<td>Array of users that ran queries per grouped variation</td>
</tr>
<tr>
<td>UNIQ_APP_NAME</td>
<td>Count of unique application names per grouped variation</td>
</tr>
<tr>
<td>ARR_APP_NAME</td>
<td>Array of application names per grouped variation</td>
</tr>
<tr>
<td>UNIQ_OS</td>
<td>Count of unique operating systems per grouped variation</td>
</tr>
<tr>
<td>ARR_OP_SYS</td>
<td>Array of operating systems per grouped variation</td>
</tr>
<tr>
<td>UNIQ_IP</td>
<td>Count of unique IPs per grouped variation</td>
</tr>
</tbody>
</table>
Query 1: Stats on Queries with Data Compression per User

Query

```sql
SELECT
    DATE(START_TIME) AS QUERY_DATE,
    1.USER_NAME AS USERNAME,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)) ,'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)) ,'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS UNIQ_OS,
    JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS') AS OP_SYS,
    COUNT( DISTINCT 1.USER_NAME) AS UNIQ_USER,
    ARRAY_UNIQUE_AGG(1.USER_NAME) AS ARR_USER,
    COUNT( DISTINCT 1.CLIENT_IP) AS UNIQ_IP,
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history ON l.EVENT_ID = s.LOGIN_EVENT_ID
WHERE
    (upper(QUERY_TEXT) LIKE '%COPY FILE%') OR
    (upper(QUERY_TEXT) LIKE '%COPY INTO%')
GROUP BY
    QUERY_DATE,
    USERNAME
```
Query 2: Stats on Queries with Data Compression per IP Address

Query

```sql
SELECT
    DATE(START_TIME) AS QUERY_DATE,
    l.CLIENT_IP AS IP,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)), 'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)), 'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS UNIQ_OS,
    JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS') AS OP_SYS,
    COUNT(DISTINCT l.USER_NAME) AS UNIQ_USER,
    ARRAY_UNIQUE_AGG(l.USER_NAME) AS ARR_USER,
    COUNT(DISTINCT l.CLIENT_IP) AS UNIQ_IP,
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history ON l.EVENT_ID = s.LOGIN_EVENT_ID
WHERE
    (upper(QUERY_TEXT) LIKE '%COPY FILE%') OR
    (upper(QUERY_TEXT) LIKE '%COPY INTO%')
GROUP BY
    QUERY_DATE,
    IP
```

© Copyright 2024 Mandiant
Query 3: Stats on Queries with Data Compression per Applications Used

```
SELECT
  DATE(START_TIME) AS QUERY_DATE,
  JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION') AS APPLICATION_NAME,
  COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
  COUNT(QRY.TEXT) AS TOTAL_QUERIES,
  TOVARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)), 'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
  TOVARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)), 'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
  COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
  ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
  COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
  ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS,
  COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS UNIQ_OS,
  JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS') AS OP_SYS,
  COUNT(DISTINCT l.USER_NAME) AS UNIQ_USER,
  ARRAY_UNIQUE_AGG(l.USER_NAME) AS ARR_USER,
  COUNT(DISTINCT l.CLIENT_IP) AS UNIQ_IP,
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history ON l.EVENT_ID = s.LOGIN_EVENT_ID
WHERE
  (upper(QRY.TEXT) LIKE '%COPY FILE%') OR
  (upper(QRY.TEXT) LIKE '%COPY INTO%')
GROUP BY
  QUERY_DATE,
  APPLICATION_NAME
```
Data Staging

Description
Analysis of queries with commands that are often associated with data staging prior to exfil.

Warnings About FP/Noise
In-house services and accounts may have back-up tasks associated with the staging commands (also with the compression commands in another section). We had to filter out for QUERY_TEXT LIKE '%NOISE%' for Kafka, spark_connector, S3, and others. Additional useful stacks to help with this can be stacking on date and OUTBOUND_DATA_TRANSFER_CLOUD, and looking for spikes in AWS/AZURE etc. and then pivoting from there.

Stacking Field Definition

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT_IP / IP</td>
<td>IP address running queries</td>
</tr>
<tr>
<td>USERNAME</td>
<td>Name of the user account that ran the query</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>The name of the application that ran queries, from CLIENT_ENVIRONMENT</td>
</tr>
<tr>
<td>EARLIEST_UTC</td>
<td>First time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>LATEST_UTC</td>
<td>Last time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>UNIQUE_QUERIES</td>
<td>Count of unique queries per grouped variation</td>
</tr>
<tr>
<td>TOTAL_QUERIES</td>
<td>Count of total queries per grouped variation</td>
</tr>
<tr>
<td>DAY_COUNT</td>
<td>Count of unique days active per grouped variation</td>
</tr>
<tr>
<td>UNIQ_USER</td>
<td>Count of unique users per grouped variation</td>
</tr>
<tr>
<td>ARR_UNIQ_USER</td>
<td>Array of users that ran queries per grouped variation</td>
</tr>
<tr>
<td>UNIQ_APP_NAME</td>
<td>Count of unique application names per grouped variation</td>
</tr>
<tr>
<td>ARR_APP_NAME</td>
<td>Array of application names per grouped variation</td>
</tr>
<tr>
<td>UNIQ_OS</td>
<td>Count of unique operating systems per grouped variation</td>
</tr>
<tr>
<td>ARR_OP_SYS</td>
<td>Array of operating systems per grouped variation</td>
</tr>
<tr>
<td>UNIQ_IP</td>
<td>Count of unique IPs per grouped variation</td>
</tr>
</tbody>
</table>
Query 1: Stats on Queries with Data Staging per User

Query

```
SELECT
    l.USER_NAME AS USERNAME,
    QUERY_TYPE,
    DATE(START_TIME) AS QUERY_DATE,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)), 'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)), 'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS,
    COUNT(DISTINCT l.CLIENT_IP) AS UNIQ_IP,
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
WHERE
    upper(QUERY_TEXT) LIKE '%ALTER STAGE%' OR
    upper(QUERY_TEXT) LIKE '%ALTER VOLUME%' OR
    upper(QUERY_TEXT) LIKE '%CREATE STAGE%' OR
    upper(QUERY_TEXT) LIKE '%CREATE VOLUME%' OR
    upper(QUERY_TEXT) LIKE '%LS %' OR
    upper(QUERY_TEXT) LIKE '%DROP %'
GROUP BY
    USERNAME
```

© Copyright 2024 Mandiant
Query 2: Stats on Queries with Data Staging per IP Address

Query

```sql
SELECT
    1.CLIENT_IP AS IP,
    QUERY_TYPE,
    DATE(START_TIME) AS QUERY_DATE,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyymmddhh24mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyymmddhh24mi:ss') AS LATEST_UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
WHERE
    (upper(QUERY_TEXT) LIKE '%ALTER STAGE%' OR
    upper(QUERY_TEXT) LIKE '%ALTER VOLUME%' OR
    upper(QUERY_TEXT) LIKE '%CREATE STAGE%' OR
    upper(QUERY_TEXT) LIKE '%CREATE VOLUME%' OR
    upper(QUERY_TEXT) LIKE '%LS %' OR
    upper(QUERY_TEXT) LIKE '%DROP %'
) GROUP BY IP
```
Query 3: Stats on Queries with Data Staging per Applications Used

Query

```sql
SELECT
    JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION') AS APPLICATION_NAME,
    QUERY_TYPE,
    DATE(START_TIME) AS QUERY_DATE,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS,
    COUNT( DISTINCT l.USER_NAME) AS UNIQ_USER,
    ARRAY_UNIQUE_AGG(l.USER_NAME) AS ARR_USER,
    COUNT( DISTINCT l.CLIENT_IP) AS UNIQ_IP,
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
WHERE
    (upper(QUERY_TEXT) LIKE '%ALTER STAGE%' OR
    upper(QUERY_TEXT) LIKE '%ALTER VOLUME%' OR
    upper(QUERY_TEXT) LIKE '%CREATE STAGE%' OR
    upper(QUERY_TEXT) LIKE '%CREATE VOLUME%' OR
    upper(QUERY_TEXT) LIKE '%LS %' OR
    upper(QUERY_TEXT) LIKE '%DROP %'
)
GROUP BY
    APPLICATION_NAME
```
Data Streams Outbound

Description

Analysis of queries to identify CREATE or ALTER STREAMS commands to detect the configuration and potential use of outbound data streams.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT_IP / IP</td>
<td>IP address running queries</td>
</tr>
<tr>
<td>USERNAME</td>
<td>Name of the user account that ran the query</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>The name of the application that ran queries, from CLIENT_ENVIRONMENT</td>
</tr>
<tr>
<td>EARLIEST_UTC</td>
<td>First time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>LATEST_UTC</td>
<td>Last time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>UNIQUE_QUERY</td>
<td>Count of unique queries per grouped variation</td>
</tr>
<tr>
<td>TOTAL_QUERY</td>
<td>Count of total queries per grouped variation</td>
</tr>
<tr>
<td>DAY_COUNT</td>
<td>Count of unique days active per grouped variation</td>
</tr>
<tr>
<td>UNIQ_USER</td>
<td>Count of unique users per grouped variation</td>
</tr>
<tr>
<td>ARR_UNIQ_USER</td>
<td>Array of users that ran queries per grouped variation</td>
</tr>
<tr>
<td>UNIQ_APP_NAME</td>
<td>Count of unique application names per grouped variation</td>
</tr>
<tr>
<td>ARR_APP_NAME</td>
<td>Array of application names per grouped variation</td>
</tr>
<tr>
<td>UNIQ_OS</td>
<td>Count of unique operating systems per grouped variation</td>
</tr>
<tr>
<td>ARR_OP_SYS</td>
<td>Array of operating systems per grouped variation</td>
</tr>
<tr>
<td>UNIQ_IP</td>
<td>Count of unique IPs per grouped variation</td>
</tr>
</tbody>
</table>
Query 1: Stats on Queries Involving Data Streams per User

Query

```sql
SELECT
    DATE(START_TIME) AS QUERY_DATE,
    l.USER_NAME AS USERNAME,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyymm-dd hh24:mi:ss') AS EARLIEST.UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyymm-dd hh24:mi:ss') AS LATEST.UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT,'APPLICATION')) AS ARR_APP_NAME,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT,'APPLICATION')) AS UNIQ_APP_NAME,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT,'OS')) AS ARR_OP_SYS
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history ON l.EVENT_ID = s.LOGIN_EVENT_ID
WHERE
    ((upper(QUERY_TEXT) LIKE '%CREATE STREAM%') OR (upper(QUERY_TEXT) LIKE '%ALTER STREAM%')) OR QUERY_TYPE IN ('CREATE_STREAM', 'ALTER_STREAM')
GROUP BY
    QUERY_DATE,
    USERNAME
```

Query 2: Stats on Queries Involving Data Streams per IP Address

Query

```sql
SELECT
    DATE(START_TIME) AS QUERY_DATE,
    l.CLIENT_IP AS IP,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
```
TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd
hh24:mi:ss') AS EARLIEST_UTC,
TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd
hh24:mi:ss') AS LATEST_UTC,
COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT,
'APPLICATION')) AS ARR_APP_NAME,
COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION'))
AS UNIQ_APP_NAME,
ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS
ARR_OP_SYS
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history ON l.EVENT_ID = s.LOGIN_EVENT_ID
WHERE
((upper(QUERY_TEXT) LIKE '%CREATE STREAM%') OR (upper(QUERY_TEXT) LIKE
'ALTER STREAM%')) OR
QUERY_TYPE IN ('CREATE_STREAM', 'ALTER_STREAM')
GROUP BY
QUERY_DATE,
IP

Query 3: Stats on Queries Involving Data Streams per Applications name

Query

Unset
SELECT
DATE(START_TIME) AS QUERY_DATE,
JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION') AS
APPLICATION_NAME,
COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd
hh24:mi:ss') AS EARLIEST_UTC,
TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd
hh24:mi:ss') AS LATEST_UTC,
COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT,
'APPLICATION')) AS ARR_APP_NAME,
Manual Data Retrieval

Description

Analysis of queries containing commands often associated with manual exfiltration of data or fetching of files that have been compressed/staged.

Warnings About FP/Noise

Check query type by stacking on filtered as GET_FILES to filter out noise, then re-run unstacked.

Stacking Field Definition

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT_IP / IP</td>
<td>IP address running queries</td>
</tr>
<tr>
<td>USERNAME</td>
<td>Name of the user account that ran the query</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>The name of the application that ran queries, from CLIENT_ENVIRONMENT</td>
</tr>
<tr>
<td>EARLIEST_UTC</td>
<td>First time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>LATEST_UTC</td>
<td>Last time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>UNIQUE_QUERIES</td>
<td>Count of unique queries per grouped variation</td>
</tr>
<tr>
<td>TOTAL_QUERIES</td>
<td>Count of total queries per grouped variation</td>
</tr>
</tbody>
</table>

© Copyright 2024 Mandiant
<table>
<thead>
<tr>
<th>DAY_COUNT</th>
<th>Count of unique days active per grouped variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIQ_USER</td>
<td>Count of unique users per grouped variation</td>
</tr>
<tr>
<td>ARR_UNIQ_USER</td>
<td>Array of users that ran queries per grouped variation</td>
</tr>
<tr>
<td>UNIQ_APP_NAME</td>
<td>Count of unique application names per grouped variation</td>
</tr>
<tr>
<td>ARR_APP_NAME</td>
<td>Array of application names per grouped variation</td>
</tr>
<tr>
<td>UNIQ_OS</td>
<td>Count of unique operating systems per grouped variation</td>
</tr>
<tr>
<td>ARR_OP_SYS</td>
<td>Array of operating systems per grouped variation</td>
</tr>
<tr>
<td>UNIQ_IP</td>
<td>Count of unique IPs per grouped variation</td>
</tr>
</tbody>
</table>

Query 1: Stats on Queries Involving Data Retrieval per User

Query

```sql
SELECT
  l.USER_NAME AS USERNAME,
  COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
  COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
  TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)) ,'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
  TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)) ,'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
  COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
  TO_NUMBER((TOTAL_QUERIES/DAY_COUNT), 10, 2) AS AVG_QUERY_PER_DAY,
  ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
  COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
  ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
```

© Copyright 2024 Mandiant
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
WHERE
  (upper(QUERY_TEXT) LIKE '%PUT FILES%')
  OR upper(QUERY_TEXT) LIKE '%GET FILES'
  OR QUERY_TYPE = 'GET_FILES'
GROUP BY
  USERNAME

Query 2: Stats on Queries Involving Data Retrieval per IP Address

Query

SELECT
  l.CLIENT_IP AS IP,
  COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
  COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
  TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
  TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
  COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
  TO_NUMBER((TOTAL_QUERIES/DAY_COUNT), 10, 2) AS AVG_QUERY_PER_DAY,
  ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
  COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
  ARRAY_UNIQUE_AGG(l.USER_NAME) AS ARR_USER,
  ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
WHERE
  (upper(QUERY_TEXT) LIKE '%PUT FILES%')
OR upper(QUERY_TEXT) LIKE '%GET FILES%'  
OR QUERY_TYPE = 'GET_FILES'  
GROUP BY  
   IP

Query 3: Stats on Queries Involving Data Retrieval per Application Name

Query

```
SELECT  
   JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION') AS APPLICATION_NAME,  
   COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,  
   COUNT(QUERY_TEXT) AS TOTAL_QUERIES,  
   TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,  
   TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,  
   COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,  
   TO_NUMBER((TOTAL_QUERIES/DAY_COUNT),10,2) AS AVG_QUERY_PER_DAY,  
   ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,  
   COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,  
   ARRAY_UNIQUE_AGG(l.USER_NAME) AS ARR_USER,  
   ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS  
FROM snowflake.account_usage.query_history q  
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id  
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID  
WHERE  
   (upper(QUERY_TEXT) LIKE '%PUT FILES%')  
OR upper(QUERY_TEXT) LIKE '%GET FILES%'  
OR QUERY_TYPE = 'GET_FILES'  
GROUP BY  
   APPLICATION_NAME
```
Network Analysis

Network Traffic Spikes

Description
Originally set to be various network-based analysis; however, after a few passes it was deemed that brute force and logon fail-rate analysis were the only high-fidelity approaches.

Warnings About FP/Noise
Later evaluation of data and discussion with Snowflake showed that values forBYTES_SENT_OVER_NETWORK and other seemingly network-based metrics are not entirely accurate. For this reason they were omitted from this section. The fields tend to be populated so you can search away as you desire; just know that those metrics should not be used as a basis for byte-count reported to client regarding exfiltration.

Stacking Field Definition

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT_IP / IP</td>
<td>IP address running queries</td>
</tr>
<tr>
<td>USERNAME</td>
<td>Name of the user account that ran the query</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>The name of the application that ran queries, from CLIENT_ENVIRONMENT</td>
</tr>
<tr>
<td>EARLIESTUtc</td>
<td>First time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>LATESTUtc</td>
<td>Last time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>UNIQUE_QUERIES</td>
<td>Count of unique queries per grouped variation</td>
</tr>
<tr>
<td>TOTAL QUERIES</td>
<td>Count of total queries per grouped variation</td>
</tr>
<tr>
<td>DAY_COUNT</td>
<td>Count of unique days active per grouped variation</td>
</tr>
<tr>
<td>UNIQ_USER</td>
<td>Count of unique users per grouped variation</td>
</tr>
<tr>
<td>ARR_UNIQ_USER</td>
<td>Array of users that ran queries per grouped variation</td>
</tr>
<tr>
<td>UNIQ_APP_NAME</td>
<td>Count of unique application names per grouped variation</td>
</tr>
<tr>
<td>ARR_APP_NAME</td>
<td>Array of application names per grouped variation</td>
</tr>
<tr>
<td>UNIQ_OS</td>
<td>Count of unique operating systems per grouped variation</td>
</tr>
<tr>
<td>ARR_OP_SYS</td>
<td>Array of operating systems per grouped variation</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>UNIQ_IP</td>
<td>Count of unique IPs per grouped variation</td>
</tr>
</tbody>
</table>

Query 1: Identify Logon Attempts (Failed and Successful)

Query

```sql
WITH FAIL_IPS AS (  
    SELECT l.CLIENT_IP AS IP  
    FROM snowflake.account_usage.login_history  
    WHERE NOT IS_SUCCESS = 'YES'  
    GROUP BY IP  
)  
SELECT  
    DATE(EVENT_TIMESTAMP) AS QUERY_DATE,  
    CLIENT_IP AS IP,  
    COUNT(DISTINCT USER_NAME) AS UNIQ_USERS,  
    COUNT_IF(IS_SUCCESS = 'YES') AS SUCCCEED_COUNT,  
    COUNT_IF(IS_SUCCESS = 'NO') AS FAIL_COUNT,  
FROM snowflake.account_usage.login_history l  
JOIN FAIL_IPS f ON f.IP = l.CLIENT_IP  
GROUP BY  
    QUERY_DATE,  
    IP
```
User Analysis

Abnormal Application Names

Description
Frequency analysis of the application names used in running queries.

Potential Attacker Trends
Combined analysis (these methodologies and others in this document) revealed the attacker using multiple applications for different stages of their attack. In our particular engagement, we observed DBeaver being used for the initial enumeration (primarily "SHOW" commands), with more fine-tuned queries executed by a .jar file, and finally large exfiltration executed by the SnowSQL app.

Warnings About FP/Noise
Snowflake has some parsing issues with the values stored in their CLIENT_ENVIRONMENT json blob. One value was stored as "C:\Program", which is believed to have resulted from parsing issues with the space in "program files". For values that parse wrong for application name, the json blob also includes versions of java/python etc. (as appropriate), and we were able to get the specific version of the jar file used by the attacker.

Stacking Field Definition

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT_IP / IP</td>
<td>IP address running queries</td>
</tr>
<tr>
<td>USERNAME</td>
<td>Name of the user account that ran the query</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>The name of the application that ran queries, from CLIENT_ENVIRONMENT</td>
</tr>
<tr>
<td>EARLIEST_UTC</td>
<td>First time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>LATEST_UTC</td>
<td>Last time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td>UNIQUE_QUERIES</td>
<td>Count of unique queries per grouped variation</td>
</tr>
<tr>
<td>TOTAL_QUERIES</td>
<td>Count of total queries per grouped variation</td>
</tr>
<tr>
<td>DAY_COUNT</td>
<td>Count of unique days active per grouped variation</td>
</tr>
<tr>
<td>UNIQ_USER</td>
<td>Count of unique users per grouped variation</td>
</tr>
<tr>
<td>ARR_UNIQ_USER</td>
<td>Array of users that ran queries per grouped variation</td>
</tr>
<tr>
<td>UNIQ_APP_NAME</td>
<td>Count of unique application names per grouped variation</td>
</tr>
<tr>
<td>ARR_APP_NAME</td>
<td>Array of application names per grouped variation</td>
</tr>
<tr>
<td>UNIQ_OS</td>
<td>Count of unique operating systems per grouped variation</td>
</tr>
<tr>
<td>ARR_OP_SYS</td>
<td>Array of operating systems per grouped variation</td>
</tr>
<tr>
<td>UNIQ_IP</td>
<td>Count of unique IPs per grouped variation</td>
</tr>
</tbody>
</table>

**Query 1: Pivot on OS**

**Query**

```
Unset
TH2-A USER, IP, OS by app name, version

SELECT
    REPORTED_CLIENT_TYPE,
    JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION') AS APPLICATION_NAME,
    SPLIT_PART(CLIENT_APPLICATION_ID, ' ', 1) AS APP_NAME_TMP,
    CASE
        WHEN REGEXP_LIKE(CLIENT_APPLICATION_ID, 'Snowflake UI .+') THEN 'N/A'
        ELSE CLIENT_APPLICATION_VERSION
    END AS APP_VERSION,
    COUNT(CLIENT_APPLICATION_VERSION) AS HIT_COUNT,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyymm-dd hh24:mi:ss') AS LATEST_UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS UNIQ_OS,
    ARRAY_UNIQUE_AGG(l.CLIENT_IP) AS ARR_IP,
    COUNT(DISTINCT 1.CLIENT_IP) AS UNIQ_IP,
    ARRAY_UNIQUE_AGG(l.USER_NAME) AS ARR_USER,
    COUNT(DISTINCT 1.USER_NAME) AS UNIQ_USER
FROM snowflake.account_usage.query_history q
```
```sql
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
GROUP BY
    REPORTED_CLIENT_TYPE,
    APPLICATION_NAME,
    APP_NAME_TMP,
    APP_VERSION
ORDER BY
    REPORTED_CLIENT_TYPE,
    APPLICATION_NAME,
    APP_NAME_TMP,
    APP_VERSION
```

Query 2: Identify/Pivot per IP

Query

```sql
SELECT
    REPORTED_CLIENT_TYPE,
    JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION') AS APPLICATION_NAME,
    SPLIT_PART(CLIENT_APPLICATION_ID, ' ', 1) AS APP_NAME_TMP,
    CASE
        WHEN REGEXP_LIKE(CLIENT_APPLICATION_ID, 'Snowflake UI .+') THEN 'N/A'
        ELSE CLIENT_APPLICATION_VERSION
    END AS APP_VERSION,
    l.CLIENT_IP AS IP_ADDR,
    COUNT(CLIENT_APPLICATION_VERSION) AS HIT_COUNT,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)), 'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)), 'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS UNIQ_OS,
    ARRAY_UNIQUE_AGG(l.USER_NAME),
COUNT( DISTINCT 1.USER_NAME) AS UNIQ_USER
FROM snowflake.account_usage.query_history q
  JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
  JOIN snowflake.account_usage.login_history l ON 1.EVENT_ID = q.LOGIN_EVENT_ID
GROUP BY
  REPORTED_CLIENT_TYPE,
  APPLICATION_NAME,
  APP_NAME_TMP,
  APP_VERSION,
  IP_ADDR
ORDER BY
  UNIQ_USER DESC

Query 3: Identify/Pivot per User

Query

SELECT
  REPORTED_CLIENT_TYPE,
  JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION') AS APPLICATION_NAME,
  SPLIT_PART(CLIENT_APPLICATION_ID, ' ', 1) AS APP_NAME_TMP,
  CASE
    WHEN REGEXP_LIKE(CLIENT_APPLICATION_ID, 'Snowflake UI .+') THEN 'N/A'
    ELSE CLIENT_APPLICATION_VERSION
  END AS APP_VERSION,
  1.USER_NAME AS USERNAME,
  COUNT(CLIENT_APPLICATION_VERSION) AS HIT_COUNT,
  TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
  TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
  COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
  ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS,
  COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS UNIQ_OS,
  ARRAY_UNIQUE_AGG(1.CLIENT_IP),
  COUNT( DISTINCT 1.CLIENT_IP) AS UNIQ_IP,
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = q.LOGIN_EVENT_ID
GROUP BY
   REPORTED_CLIENT_TYPE,
   APPLICATION_NAME,
   APP_NAME_TMP,
   APP_VERSION,
   USERNAME
ORDER BY
   UNIQ_IP DESC

Average User Sessions

Description

Sessions are tied to individual user logons, so multiple logins for any given username will generate multiple session IDs. Given that most client services are crafted to login and execute a specific purpose on a recurring basis, an attacker may logon multiple times for various different reasons. Sessions per username spikes may indicate anomalous use.

Warnings About FP/Noise

A count of unique session IDs per IP per day can also show meaningful data for analysis. Care should be taken as to the volume of IPs in your dataset before proceeding.

Running a distinct count of IPs, or distinct IPs per day/week/month, can help identify whether your IP volume will be problematic for an extended query or not. An additional WHERE filter to exclude all IPs where distinct session ID count <2 can also remove high volumes.

Stacking Field Definition

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT_IP / IP</td>
<td>IP address running queries</td>
</tr>
<tr>
<td>USERNAME</td>
<td>Name of the user account that ran the query</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>The name of the application that ran queries, from CLIENT_ENVIRONMENT</td>
</tr>
<tr>
<td><strong>EARLIEST_UTC</strong></td>
<td>First time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td><strong>LATEST_UTC</strong></td>
<td>Last time, grouped variation was observed, normalized to UTC as a string</td>
</tr>
<tr>
<td><strong>UNIQUE_QUERIES</strong></td>
<td>Count of unique queries per grouped variation</td>
</tr>
<tr>
<td><strong>TOTAL_QUERIES</strong></td>
<td>Count of total queries per grouped variation</td>
</tr>
<tr>
<td><strong>DAY_COUNT</strong></td>
<td>Count of unique days active per grouped variation</td>
</tr>
<tr>
<td><strong>UNIQ_USER</strong></td>
<td>Count of unique users per grouped variation</td>
</tr>
<tr>
<td><strong>ARR_UNIQ_USER</strong></td>
<td>Array of users that ran queries per grouped variation</td>
</tr>
<tr>
<td><strong>UNIQ_APP_NAME</strong></td>
<td>Count of unique application names per grouped variation</td>
</tr>
<tr>
<td><strong>ARR_APP_NAME</strong></td>
<td>Array of application names per grouped variation</td>
</tr>
<tr>
<td><strong>UNIQ_OS</strong></td>
<td>Count of unique operating systems per grouped variation</td>
</tr>
<tr>
<td><strong>ARR_OP_SYS</strong></td>
<td>Array of operating systems per grouped variation</td>
</tr>
<tr>
<td><strong>UNIQ_IP</strong></td>
<td>Count of unique IPs per grouped variation</td>
</tr>
</tbody>
</table>

### Query 1: Stack by User

**Query**

```
SELECT l.USER_NAME AS USERNAME,
       DATE(START_TIME) AS QUERY_DATE,
       COUNT(DISTINCT s.SESSION_ID) AS UNIQ_SESSIONS,
       COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
       COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
       TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
       TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
       COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
       ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT,'APPLICATION')) AS ARR_APP_NAME,
       COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT,'APPLICATION')) AS UNIQ_APP_NAME,
```
QUERY 2: STACK BY IP

SELECT
    1.CLIENT_IP AS IP,
    DATE(START_TIME) AS QUERY_DATE,
    COUNT(DISTINCT s.SESSION_ID) AS UNIQ_SESSIONS,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)) , 'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS
FROM snowflake.account_usage.query_history q
JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = s.LOGIN_EVENT_ID
GROUP BY
    QUERY_DATE,
    IP

© Copyright 2024 Mandiant
Query 3: Stack by App

Query

```sql
SELECT
    JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION') AS APPLICATION_NAME,
    DATE(START_TIME) AS QUERY_DATE,
    COUNT(DISTINCT s.SESSION_ID) AS UNIQ_SESSIONS,
    COUNT(DISTINCT QUERY_HASH) AS UNIQUE_QUERIES,
    COUNT(QUERY_TEXT) AS TOTAL_QUERIES,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS EARLIEST_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', START_TIME)),'yyyy-mm-dd hh24:mi:ss') AS LATEST_UTC,
    COUNT(DISTINCT DATE(START_TIME)) AS DAY_COUNT,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS ARR_APP_NAME,
    COUNT(DISTINCT JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'APPLICATION')) AS UNIQ_APP_NAME,
    ARRAY_UNIQUE_AGG(JSON_EXTRACT_PATH_TEXT(s.CLIENT_ENVIRONMENT, 'OS')) AS ARR_OP_SYS
FROM snowflake.account_usage.query_history q
    JOIN snowflake.account_usage.sessions s ON s.session_id = q.session_id
    JOIN snowflake.account_usage.login_history l ON l.EVENT_ID = s.LOGIN_EVENT_ID
GROUP BY
    QUERY_DATE,
    APPLICATION_NAME
```

© Copyright 2024 Mandiant
Query for Temporary Stages and External URLs

Description

Unlike the majority of the tables in “snowflake.account_usage,” the “stages” view will show a table of all stages (Internal and External) historically created in that particular Snowflake instance. Columns include timestamps showing creation date, modification date, and deletion date, but most importantly the URL used to connect to a cloud environment if the stage was external. A review of these distinct stage URLs can quickly identify unauthorized connections previously or currently being made in the Snowflake environment.

The below query pulls out all external URLs used for stages. STAGE_TYPE can include AWS, AZURE, etc. These can then be validated with the client to look for rogue stages.

Note that this table also includes the names of stages that have been deleted. If the attacker creates and deletes a stage, the client will not necessarily be able to recover the stage, but there will be evidence of the creation/deletion of the stage and the URL it was pointed to.

Query 1: Stage Name URL Where URL Is not NULL

Query

```
SELECT
    STAGE_NAME,
    STAGE_SCHEMA,
    STAGE_CATALOG,
    STAGE_URL,
    STAGE_REGION,
    STAGE_TYPE,
    STAGE_OWNER,
    TO_VARCHAR(MIN(CONVERT_TIMEZONE('UTC', CREATED)) , 'yyyymm-dd hh24:mi:ss') AS CREATE_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', LAST_ALTERED)) , 'yyyymm-dd hh24:mi:ss') AS LAST_ALTERED_UTC,
    TO_VARCHAR(MAX(CONVERT_TIMEZONE('UTC', DELETED)) , 'yyyymm-dd hh24:mi:ss') AS DELETED_UTC,
    COUNT(DISTINCT STAGE_ID) AS UNIQ_STAGE_ID,
    ARRAY_UNIQUE_AGG(STAGE_ID) AS ARR_STAGE_ID
FROM snowflake.account_usage.stages
WHERE
    STAGE_URL IS NOT NULL
GROUP BY
```

© Copyright 2024 Mandiant
| STAGE_NAME, |
| STAGE_SCHEMA, |
| STAGE_CATALOG, |
| STAGE_URL, |
| STAGE_REGION, |
| STAGE_TYPE, |
| STAGE_OWNER |