Setting the Stage
WHAT IS SNOWFLAKE?

Snowflake is a SaaS data platform

Customers:
• Create and manage users
• Load data
• Execute commands
• Export data

The Snowflake Service:
• Processes requests
• Maintains security
• Manages capacity

Snowflake personnel do not have access to unencrypted customer data

Snowflake personnel do not collect, delete, update, disclose, or use customer data

• Orchestrated security built into the fabric of the platform
  • Automated controls in place for all functions
  • Constant monitoring
  • Analysis to detect and mitigate threats quickly

Built-in security features protect the data you load and use in Snowflake

Snowflake uses the industry-standard “shared responsibility” model

Snowflake uses sophisticated mechanisms to keep the platform safe and stable
SECURE FOUNDATION

- Customers never have direct access (e.g., "SSH") to the Snowflake VPC/VNET
- All access to Customer Data is through the Snowflake Service application layer
- Customer Data is decrypted only in memory on dedicated Virtual Warehouse VMs
  - Only the data **required to process the command** is decrypted
  - Virtual Warehouses are **ephemeral**, meaning they run only when needed

Customer Data is encrypted at rest using **dedicated encryption keys**
Storage is governed by **dedicated IaaS IAM users**

Secure traffic between all points
(Encrypted at rest and in transit)

Tri-Secret Secure Customer Managed Keys
(Bring Your Own Key enabled)

Virtual Warehouses are **dedicated** to each Customer
SNOWFLAKE SECURITY AT A GLANCE

Operational Controls
- NIST 800-53
- SOC2 Type 2
- ISO 27001

Access
- All communication secured & encrypted
- TLS 1.2 encryption for all client communications
- IP Allow List

Authentication
- User Provisioning with SCIM
- Password Policy enforcement
- Multifactor Authentication
- SSO using SAML 2.0 Federated Authentication
- Key Pair (link to snowsql, but supported all over)

Authorization
- RBAC for data and actions
- OAuth2.0 authorization framework support
- Secure views and UDFs to protect information access
- Full audit trails for every action any user does

Data
- Encrypted at rest
- Hierarchical key model rooted in AWS Cloud HSM or Azure Dedicated HSM
- Automatic key rotation
- Time Travel 1 day
- Fail Safe

Infrastructure
- AWS, Azure Physical Security
- AWS, Azure Redundancy
- Regional Data Centers
  - US
  - EU
  - AP
SNOWFLAKE SECURITY AT A GLANCE

Access
- All communication secured & encrypted
- TLS 1.2 encryption for all client communications
- IP Allow List

Authentication
- User Provisioning with SCIM
- Password Policy enforcement
- Multifactor Authentication
- SSO using SAML 2.0 Federated Authentication
- Key Pair (link to snowsql, but supported all over)

Authorization
- RBAC for data and actions
- OAuth2.0 authorization framework support
- Secure views and UDFs to protect information access
- Full audit trails for every action any user does

Data
- Encrypted at rest
- Hierarchical key model rooted in AWS Cloud HSM or Azure Dedicated HSM
- Automatic key rotation
- Yearly re-keying
- Time Travel 1-90 days
- Fail Safe

Infrastructure
- AWS, Azure Physical Security
- AWS, Azure Redundancy
- Regional Data Centers
  - US
  - EU
  - AP

Operational Controls
- NIST 800-53
- SOC2 Type 2
- ISO 27001
SNOWFLAKE SECURITY AT A GLANCE
Business Critical Edition

Access
- All communication secured & encrypted
- TLS 1.2 encryption for all client communications
- Option for encryption in both trusted and untrusted networks
- IP Allow List
- Support for Azure Cross VNET rules for Blob access
- Integration with AWS Private Link
- Integration with Azure Private Link

Authentication
- User Provisioning with SCIM
- Password Policy enforcement
- Multifactor Authentication
- SSO using SAML 2.0 Federated Authentication
- Key Pair (link to snowsql, but supported all over)

Authorization
- Flexible & granular authorization controls
- RBAC for data and actions
- OAuth2.0 authorization framework support
- Secure views and UDFs to protect information access
- Full audit trails for every action any user does

Data
- Encrypted at rest
- Hierarchical key model rooted in AWS Cloud HSM or Azure Dedicated HSM
- Automatic key rotation
- Yearly re-keying
- Tri-Secret Secure (BYOK)
- Time Travel 1-90 days
- Fail Safe

Infrastructure
- AWS, Azure, GCP Physical Security
- AWS, Azure, GCP Redundancy
- Regional Data Centers
  - US
  - EU
  - AP

Snowflake Operational Controls
- FedRAMP / NIST 800-53
- HIPAA / HITRUST
- PCI
- ISO/IEC 27001
- SOC2 Type 2
- SOC1 Type 2

© 2020 Snowflake Inc. All Rights Reserved
Acess, Authentication & Authorization
Common Connection Pattern for Drivers & Connectors

- Every driver & connector connects the same way
- All communication encrypted end-to-end
  - All customer data flows solely over HTTPS
  - Connections encrypted using TLS 1.2 from client through to the Snowflake Service
- Data encrypted at rest

Common Access Control for all Sessions

- IP allowlisting available to restrict client communication to specified IP addresses using customer-configured Network Policies
- Authentication required for all connections
ACCESS – NETWORK POLICIES

Network Policies
- Encapsulate an IP allowlist and an IP blocklist
- Customer-configured Network Policy

Network policies can be applied at three levels

1. Snowflake Account
   - All traffic will use this policy, unless there is a more specific one.
   - Control is applied at authentication time.

2. Outside Integration
   - Applies to traffic at the integration endpoint only.
     For example: SCIM or OAuth security integrations

3. User Specific
   - Applies to the specific user only.
   - Best practice for users used as service accounts.

The most specific policy always wins.
USER PROVISIONING WITH SCIM

Authoritative Directory in Control

- User creation, changes, & deletes
- Roles driven by group membership
- Use Okta, Azure AD, or any system that speaks SCIM, or...
- Any system that can use SQL
There are three scenarios when you want to authenticate to Snowflake

1. **Built-in user authentication**
   - This means there is a password in Snowflake, and the user authenticates with Snowflake
   - You can use MFA with this, but it only implements DUO, so the user needs a DUO token
   - Users are subject to the Snowflake Password Policy

2. **SSO for human, interactive use cases**
   - This can use SAML-based SSO with any SAML 2 compatible IdP
   - Some partner applications also deliver this leveraging Snowflake OAuth or External OAuth
   - “External Browser” – On a desktop system, the driver pops open a browser for the user to log in

3. **Authentication for non-human users (programmatic, service account, etc.)**
   - Built-in passwords may be used for this but is not recommended
   - Key Pair authentication can be used and is best paired with a secrets management solution
   - External OAuth can be leveraged here, and is the only method that will allow for an SSO-based user credential in the programmatic scenario
Authorization Control

- Role-based authorization
- Authorization for access to all database objects—databases, schemas, tables...
- Authorization for operations in Snowflake—create, stop & start virtual warehouses
- DAC and RBAC info

Application Auditing

- All actions are logged
- Audit Logs available through Snowflake Service

Diagram:

Users

Role 1
- View tables X, Y
- Use warehouse A

Role 2
- View table Y
- Modify table Y
- Use warehouse B
Information in Original Form

- Loaded in its original form
- Protected by RBAC, structure, and policy
- Built-in Snowflake Admin roles control access policy

At query time:

- Redact rows, columns, or specific results
- Apply DDM or any Snowflake function in all queries and shares using policy-driven protection

Information Stored in Tables

<table>
<thead>
<tr>
<th>Table A</th>
<th>Table B</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>971</td>
</tr>
<tr>
<td>Alice</td>
<td>A****</td>
</tr>
<tr>
<td>FE</td>
<td>oidhe98</td>
</tr>
<tr>
<td>456</td>
<td>823</td>
</tr>
<tr>
<td>Bob</td>
<td>B****</td>
</tr>
<tr>
<td>NB</td>
<td>xcnab23</td>
</tr>
<tr>
<td>789</td>
<td>467</td>
</tr>
<tr>
<td>Chaz</td>
<td>C****</td>
</tr>
<tr>
<td>TG</td>
<td>ncmbc21</td>
</tr>
</tbody>
</table>

All data in Table A & B is protected with encryption at rest with optional Tri-Secret Secure (BYOK).

Information in Manipulated Form

- Loaded in a protected form
- Also protected by RBAC, structure, and policy
- 3rd-party system(s) required to change access policy

At query time:

- Results are clear or not clear depending on 3rd-party managed policies

COLUMN-LEVEL SECURITY, SECURE VIEWS, & UDFs

Example:

create or replace masking policy protect_it as (val string) returns string ->
  case
  when is_granted_to_invoker_role('PAYROLL')
    then val
  when is_granted_to_invoker_role('ANALYST')
    then regexp_replace(val, '[0-9]', '*', 7)
  else ** masked **
  end;

Functions:

- Any built-in function or UDF
- Text:
  - regexp_replace(), concat(), etc.
- Obfuscation:
  - hash()
  - encrypt() & decrypt()
  - encrypt_raw() & decrypt_raw()
- Any External Function

* currently in preview
Encryption
Hierarchical Key Model

- Hierarchical key model rooted in Dedicated HSM
- Encrypts data at rest
- Data is encrypted by default, with no configuration required
Hierarchical Key Model using Tri-Secret Secure

- Hierarchical key model rooted in Dedicated HSM
- Encrypts data at rest
- Tri-Secret Secure
- Data is encrypted by default, with no configuration required
KEY ROTATION & RE-KEYING

Key Rotation
- Snowflake rotates AMK & TMK every 30 days
- Process is transparent to customer and queries

Key Re-Keying
- Yearly re-keying re-encrypts data on the key's birthday
- Re-keying requires Enterprise Edition or better
- Process is transparent to customer and queries

More resources on Key Management
TIME TRAVEL & FAIL SAFE

Continuous Data Protection Lifecycle

Standard operations allowed:
Queries, DDL, DML, etc.

Time Travel allowed:
- SELECT ... AT | BEFORE ...
- CLONE ... AT | BEFORE ...
- UNDROP ...

No user operations allowed
(data recoverable only by Snowflake)

Current Data Storage
Time Travel Retention (1-90 Days)
Fail-Safe (transient: 0 days, Permanent: 7 days)

Time Travel
- Select from data as it existed in the past, e.g. before some specific event
- Up to 90 days

Fail Safe
- Request recovery of lost data
- Up to 7 days for most objects

More on Time Travel & Fail Safe
Application Logging & Monitoring
APPLICATION ACTIVITY LOGGING

History Tab
- Provides a list of all commands
- Clicking on any item brings up additional details about the action
- For errors, provides SQL as well as a verbose error statement
The LOGIN_HISTORY family of table functions can be used to query login attempts by Snowflake users along various dimensions. More information can be found in our [LOGIN_HISTORY](#) documentation.

- **LOGIN_HISTORY** – returns login events within a specified time range
- **LOGIN_HISTORY_BY_USER** – returns login events of a specified user within a specified time range
- Results can be further filtered using SQL predicates
- Can be used to return login activity for up to 7 days in the past
- Export as JSON for use in SIEM or Snowflake

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENT_TIMESTAMP</td>
<td>TIMESTAMP_LTZ</td>
<td>Time of the event occurrence.</td>
</tr>
<tr>
<td>EVENT_ID</td>
<td>NUMBER</td>
<td>Event's unique id.</td>
</tr>
<tr>
<td>EVENT_TYPE</td>
<td>TEXT</td>
<td>Event type, such as LOGIN for authentication events.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>TEXT</td>
<td>User associated with this event.</td>
</tr>
<tr>
<td>CLIENT_IP</td>
<td>TEXT</td>
<td>IP address where the request originated from.</td>
</tr>
<tr>
<td>REPORTED_CLIENT_TYPE</td>
<td>TEXT</td>
<td>Reported type of the client software, such as JDBC_DRIVER, ODBC_DRIVER, etc. This information is not authenticated.</td>
</tr>
<tr>
<td>REPORTED_CLIENT_VERSION</td>
<td>TEXT</td>
<td>Reported version of the client software.</td>
</tr>
<tr>
<td>FIRST_AUTHENTICATION_FACTOR</td>
<td>TEXT</td>
<td>Method used to authenticate the user (the first factor, if using multi factor authentication).</td>
</tr>
<tr>
<td>SECOND_AUTHENTICATION_FACTOR</td>
<td>TEXT</td>
<td>The second factor, if using multi factor authentication, or NULL otherwise.</td>
</tr>
<tr>
<td>IS_SUCCESS</td>
<td>TEXT</td>
<td>Whether the user’s request was successful or not.</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>NUMBER</td>
<td>Error code, if the request was not successful.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>TEXT</td>
<td>Error message returned to the user, if the request was not successful.</td>
</tr>
<tr>
<td>RELATED_EVENT_ID</td>
<td>NUMBER</td>
<td>Reserved for future use.</td>
</tr>
</tbody>
</table>
exporting query logs – query_history

- **query_history** returns queries within a specified time range.

- **query_history_by_session** returns queries within a specified session and time range.

- **query_history_by_user** returns queries submitted by a specified user within a specified time range.

- **query_history_by_warehouse** returns queries executed by a specified warehouse within a specified time range.

- Up to 7 days

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query_id</td>
<td>text</td>
<td>The statement's unique id.</td>
</tr>
<tr>
<td>query_text</td>
<td>text</td>
<td>Text of the SQL statement.</td>
</tr>
<tr>
<td>query_type</td>
<td>text</td>
<td>DML, query, etc. If the query is currently running, or the query failed, then the query type may be unknown.</td>
</tr>
<tr>
<td>session_id</td>
<td>number</td>
<td>Session that executed the statement.</td>
</tr>
<tr>
<td>user_name</td>
<td>text</td>
<td>User who issued the query.</td>
</tr>
<tr>
<td>role_name</td>
<td>text</td>
<td>Role that was active in the session at the time of the query.</td>
</tr>
<tr>
<td>database_name</td>
<td>text</td>
<td>Database that was in use at the time of the query.</td>
</tr>
<tr>
<td>schema_name</td>
<td>text</td>
<td>Schema that was in use at the time of the query.</td>
</tr>
<tr>
<td>warehouse_name</td>
<td>text</td>
<td>Warehouse that the query executed on, if any.</td>
</tr>
<tr>
<td>warehouse_size</td>
<td>text</td>
<td>Size of the warehouse when this statement executed.</td>
</tr>
<tr>
<td>warehouse_type</td>
<td>text</td>
<td>Type of the warehouse when this statement executed.</td>
</tr>
<tr>
<td>query_tag</td>
<td>text</td>
<td>Query tag set for this statement through the query_tag session parameter.</td>
</tr>
<tr>
<td>execution_status</td>
<td>text</td>
<td>running, queued, blocked, success, error, incident, ...</td>
</tr>
<tr>
<td>error_code</td>
<td>number</td>
<td>Error code, if the query returned an error</td>
</tr>
<tr>
<td>error_message</td>
<td>text</td>
<td>Error message, if the query returned an error</td>
</tr>
<tr>
<td>start_time</td>
<td>timestamp</td>
<td>Statement's start time.</td>
</tr>
<tr>
<td>end_time</td>
<td>timestamp</td>
<td>Statement's end time, or NULL if the statement is still running.</td>
</tr>
<tr>
<td>total_elapsed_time</td>
<td>number</td>
<td>Elapsed time (in milliseconds)</td>
</tr>
<tr>
<td>compilation_time</td>
<td>number</td>
<td>Compilation time (in milliseconds)</td>
</tr>
<tr>
<td>execution_time</td>
<td>number</td>
<td>Execution time (in milliseconds)</td>
</tr>
<tr>
<td>queued_provisioning_time</td>
<td>number</td>
<td>Time (in milliseconds) spent in the warehouse queue, waiting for the warehouse servers to provision, due to warehouse creation, resume, or resize.</td>
</tr>
<tr>
<td>queued_repair_time</td>
<td>number</td>
<td>Time (in milliseconds) spent in the warehouse queue, waiting for servers in the warehouse to be repaired.</td>
</tr>
<tr>
<td>queued_overload_time</td>
<td>number</td>
<td>Time (in milliseconds) spent in the warehouse queue, due to the warehouse being overloaded by the current query workload.</td>
</tr>
<tr>
<td>transaction_blocked_time</td>
<td>number</td>
<td>Time (in milliseconds) spent blocked by a concurrent DML.</td>
</tr>
</tbody>
</table>
EXPORTING USER ACCESS AUDIT LOGS

- History logs via functions and views
  - LOGIN_HISTORY
  - QUERY_HISTORY
- Results can be further filtered using SQL predicates
- Can be used to return login activity for up to 7 days in functions 365 days in views in the past
- Export through JDBC or as JSON for use in SIEM
- All supplied drivers and connectors also have extended logging
Infrastructure Security
How is the Snowflake Infosec Team monitoring the service?

Snowflake’s internal Critical Security Controls dashboard provides real-time risk visibility

- Access Control, Security Assessment & Authorization, Configuration Management, Security Awareness, etc. all represented on a single Dashboard

- Real-time monitoring of data loaded into Snowflake from internal and other relevant data sources

Snowflake uses CIS benchmark templates for configuration hardening

- Service configuration information is collected centrally in Snowflake

- Continuously and automatically tracked—unplanned changes cause alerts

- Part of the Snowflake Security Compliance Team’s dashboard

Snowflake performs 7–12 3rd-party penetration tests per year

- Comprehensive Web Application Penetration Test – Annually

- Internal Network Penetration Test – Annually

- Major Functionality Penetration Tests – As major functionality is released as part of the SDLC

Snowflake performs weekly vulnerability scans on infrastructure

- Vulnerabilities are remediated per Security Policy

- Remediation trends tracked using Snowflake
SECURITY ANALYTICS WAREHOUSE

Snowflake Internal

Security Engineers

Internet Registration Trackers

“Known Good” Trackers

Indicators of Compromise
TTPs and Attribution
Hash Whitelists
Detection Queries
Suppression Queries
Security Alerts
Raw Events

Threat intelligence Vendors

Security Solution Providers
Query examples:
AWS login without MFA?
Customer bucket / blob data modified?
Unauthorized install on server?
Grant permissions on Snowflake?

Log Data Table
Query Table
Alerts Table
INFRASTRUCTURE LOGGING & MONITORING

Snowflake’s Security Analytics Warehouse

SnowAlert provides a security analytics framework

OpenSource, [hosted on GitHub]

Integrates with other systems for notification and tracking (Slack, JIRA, etc.)

Provides real-time data and trends

SnowAlert and sample alert to detect the startup of shell scripts in the EC2 environment
SNOWALERT – SAMPLE ALERTS

Display a dashboard of actionable alerts with links to issue-tracking systems

Show trends to measure improvement, monitor problem areas
Compliance Frameworks
INFOSEC & COMPLIANCE
All reports, attestations, documentation, and certifications

Third-Party Reports & Certifications
- Snowflake SOC 2 Type II Report
- Snowflake SOC 1 Type II Report
- Snowflake PCI-DSS-AOC-Final Report
- HIPAA Report (proving ability to enter BAA)
- HITRUST Certified
- Snowflake’s ISO 27001 Certificate
- FedRAMP Moderate (on OMB MAX)
- CyberGRX Report
- Penetration Test Results

Snowflake’s Policy Documentation
- Snowflake Security Policy
- https://www.snowflake.com/legal/ for Acceptable Use, Support, and more

Snowflake Internal Controls & Testing
- DRP, BCP, and Information System Contingency Plans
- Security Incident Process
- Staff Training, Onboarding, and Access Policies

Snowflake Self-Assessment Reports
- CAIQ
- SIG Lite
- Red Team Pen Tests

SOC 2 Type II
12 Month Coverage Period

SOC 1 Type II
6 Month Coverage Period
GDPR – GENERAL DATA PROTECTION REGULATION

What it is

- GDPR is a new EU regulation that becomes effective on May 25, 2018
- Governs the protection and processing of EU personal data

What it means in the context of Snowflake

Different requirements apply to different types of entities

- Controller – Snowflake Customers are responsible for complying with GDPR independently from Snowflake
- Processor – Snowflake is responsible for the following:
  - Putting data processing addendums in place with our customers and our vendors
  - Only using our customers' EU personal data to provide our service to them
  - Being transparent about how we handle and process our customers' EU personal data on their behalf and keeping accurate records
  - Securing customers' EU personal data in our service
  - Facilitating our customers' compliance with data subject requests
  - Notifying customers about changes to our list of subcontractors

Snowflake responsibilities are documented in a Data Processing Addendum (DPA on snowflake.com/legal)

Available for signature now
ADDITIONAL COLLATERAL

Snowflake Security Product Documentation

The above link provides information on how to configure:

- Network Policies
- MFA, IP Whitelisting
- Federated Authentication / SSO
- Access Control (DAC, RBAC)
- Best Practices
- Audit Logs
THANK YOU
Appendix
SNOWALERT: SECURITY ANALYTICS AT A GLANCE

DATA SOURCES

- Semi-Structured Data
  - Anti-Malware
  - Email
  - Web

- Network Flows
- Operating System
- Database

- Cloud Data
  - File Sharing
  - SaaS / IaaS data and logs
  - Storage Buckets

- On-Premise Structured Data
  - Database
  - Data Warehouse
  - AD, CMDB, threat intel

ANALYTICS / ORCHESTRATION

- Ticketing / SIEM
  - splunk
  - Jira Software
  - ArcSight
  - pagerduty
  - elastic
  - servicenow

- BI / Data Science
  - tableau
  - Looker
  - Sigma
  - alteryx

- Operating System
- Email
- Network Flows
- Anti-Malware
- Ticketing / SIEM

- We used to use 3rd party SIEM and it didn’t scale
- We built our new solution on Snowflake itself
- We took the SnowAlert layer and put it on Github
- We want everyone to be able to harness the power we use ourselves
SNOWFLAKE AUTHENTICATION

Five ways to authenticate:

1. **Built-in username/password** – Password natively stored in Snowflake USER object. USER object delivered as a string or typed in by the user. May be the only option for tools that only authenticate with a username and password.

2. **RSA key pair** – The client uses its private key to encrypt a string (the *fingerprint*), and Snowflake uses one of the app’s two public keys to decrypt it and authenticate.

3. **SSO powered by SAMLv2** – Federated single sign-on. Three identity providers fully support this method: Okta, AzureAD, ADFS. Other identity providers do not support single logout.

4. **OAuth 2.0 code grant flow** – SSO from server-based client applications. Requires client application modifications.
   - Snowflake OAuth uses Snowflake as the authorization service (authenticating to Snowflake through one of the other methods).
   - External OAuth leverages the customer’s authorization service. This is the only method that will allow for an SSO-based user credential in the programmatic scenario. Requires client application modifications.

5. **“External Browser”** – On a desktop system, the Snowflake driver pops open the system’s browser for the user to log in.
   - Most often used for SAMLv2-based SSO authentication but works with built-in username/password as well.
   - Requires end-user intervention and only works with desktop client applications.

Two main authentication use cases:

- Authenticating human users.
- Authenticating service accounts used by applications.
SNOWFLAKE WITH AWS PRIVATELINK AND AWS DIRECT CONNECT

Amazon Region

- Snowflake Multi-Tenant Pod
  - Load Balancer (NLB)
  - Cloud Services
  - Metadata Store
  - Virtual Warehouses

Secure Traffic within Region

Customer Environment (VPC) within AWS

- Secure traffic over the Internet (Encrypted at Rest and in Transit)
- Tri-Secret Secure Customer Managed Keys (Bring Your Own Key Enabled)
- Customer Dedicated Virtual Instances
- AWS PrivateLink

Customer Corporate Environment
How Snowflake Integrates with AWS PrivateLink

Each customer is uniquely configured with an NLB in their region.

PrivateLink is often a piece of a larger private comms requirement.