SQL Driven Infrastructure for Cybersecurity ML Operations

Dr. Konstantin Berlin
CAMLIS 2023

Image by DALL-E 3
Who am I and why am I talking about this?

• Former Head of AI at Sophos
• Worked on the cyber MLOps problem for 7+ years
• Bad infra is the main problem for most AI teams
• Good infra lets you do great work: @CAMLIS2023
  • Web content filtering through knowledge distillation of Large Language Models – Tamas Voros (tomorrow)
  • Playing Defense: Benchmarking Cybersecurity Capabilities of Large Language Models - Adarsh Kyadige (tomorrow)
• Views are my own
Observe, Orient, Decide, Act (OODA) Loop In Cybersecurity

- **Deploy security rules**
  - ML models
  - Human rules
  - Cloud lookups

- **Collect intelligence**
  - Monitor detections
  - Monitor misses
  - Collect 3rd party information

- **Correlate results**
  - Identify false positive
  - Identify misses
  - Integrate 3rd party information with customer telemetry

- **Update detection rules**
  - Update ML models
  - Create/update human rules
  - Change how ML and rules are combined

- **This is war**
- Iteration time between new detection logic releases determines company success
- The more information taken into the decision the more attacks can be found
- Releasing new detection capabilities requires collaboration across teams
- Infrastructure silos prevent experimentation and rapid deployment of new tech
- More complex detection logic being added to the cloud (XDR/MDR)
Current **Imperative Cyber Pipelines**

- **Decorations** have limited access to data
- **Detections** have limited context
- Dependency between components
- Hard to run multiple versions at the same time
- **Data Science** visibility into the pipeline low
- Skewed distributions in DS
- DS has no agency
  - Difficult to get new data
  - Impossible to test before deployment
- DS wrangling capacity limited to a single notebook or requires engineering support
- New flows take months/years to deploy
- Super expensive

**Major Issues**

- Decorations have limited access to data
- Detections have limited context
- Dependency between components
- Hard to run multiple versions at the same time
- DS visibility into the pipeline low
- Skewed distributions in DS
- DS has no agency
  - Difficult to get new data
  - Impossible to test before deployment
- DS wrangling capacity limited to a single notebook or requires engineering support
- New flows take months/years to deploy
- Super expensive
Cloud Native Modern **Declarative** SQL
Warehouses: Snowflake, Databricks, Iceberg, etc.

- SQL pipelines are more declarative
  - You define the what
  - Warehouse decided the how automatically
- Previously the automatic how was very bad
  - A lot of hacking of the how (imperative)
- Now the automatic how is approaching the manual imperative approaches
  - Storage based on massively scalable data lakes (ex. S3)
  - Data management layer removes a lot of the how
- Scaling and speed approaches imperative systems
  - Compute separate from storage
  - On-demand infinite compute
  - Resource independent workflows
  - Streaming support
- All the data in one place
  - Quick development
- Semi-structured data support
- Build the declaratively pipeline optimize later

Declarative SQL Driven Infrastructure

- Pipeline purely logical
- Deployment and dev is unified
- New pipelines easy to test and deploy
- Same scalability as in production

Example:

```sql
SELECT url, url_model(url_features(url)) as score
FROM mdr WHERE type='url_scan'
```
URL Pipeline: Legacy Setup

- **Internal Telemetry**
- **External Intelligence**
- **Decorations**
  - Python based
  - Normalizes semi-structured data
- **Aggregations**
  - SQL based
  - Requires normalization
- **Model Scoring**
  - Python based
  - Provided on ingestion
- **Training**
  - EC2 Local data processing
  - Requires feature extraction

`http://g.com/search?q=♥`
`http://g.com/search?q=%E2%99%A5`
URL Pipeline: Legacy Setup

Internal Telemetry

External Intelligence

Decorations
- Python based
- Normalizes semi-structured data

Model Scoring
- Python based
- Provided on ingestion

Aggregations
- SQL based
- Nightly
- Incremental
- Requires decorations

Training
- EC2 Local data processing
- Incremental
- Requires decorations
URL Pipeline: Snowflake Centric Setup

Key Snowflake Tech:
• Snowpipe
• Semi-structured data support
• Python UDFs
• Python Snowpark

Ingestion
• Snowpipe
• Data ingested in raw, semi-structured form

Aggregations
• Python UDFs
• Containers Supporting Snowflake API

Model Scoring & Decorations
• External Functions
• Python UDFs

Training
• Python Snowpark
• Export to AWS Sagemaker

Internal Telemetry
External Intelligence
Observed Benefits

• Significantly improved our OODA loop iteration time
• Unifying deployment and development platforms
• Simplified data wrangling by giving data scientists an easy-to-use distributed compute platform
• Provided cross-team access to all key pieces of detection tech within an easy-to-use SQL interface
• Allowed data scientists to directly develop models on top of “raw” semi-structured data input
“Electric castle in space that has 100 dogs in it.”
Image generated by Emmanuel Berlin