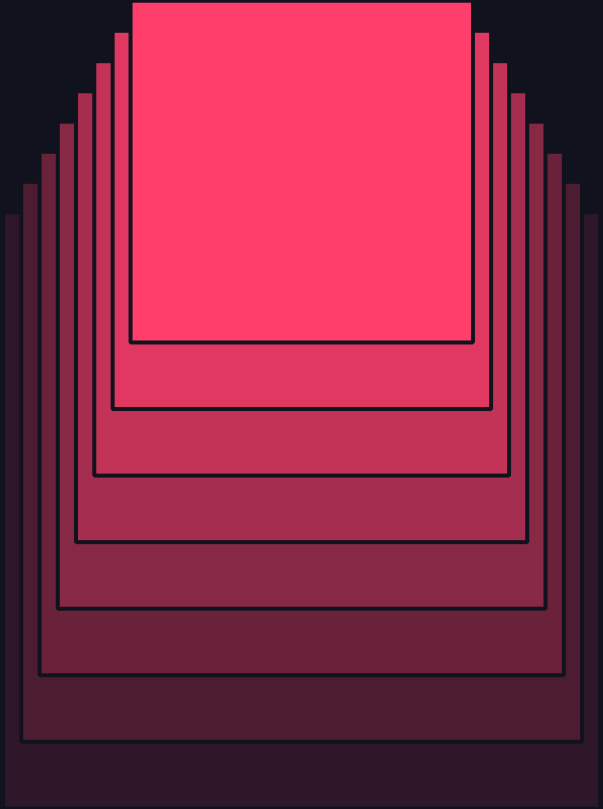
DATA+AI SUMMIT BY databricks

CYBERSECURITY DATA PLATFORM AT RIVIAN



Chris Mandich & Rob Hartman
June 13, 2024

RIVIAN CYBERSECURITY AND OUR DATA JOURNEY

Overview of Rivian

"Keep The World Adventurous Forever"

We are an American electric vehicle maker focused on adventure and sustainability. Our vehicles are designed for versatility and off-road capability. We are committed to renewable manufacturing and reducing environmental impact for the world.

Rivian Cybersecurity

Who are we?

- The Rivian Cybersecurity Operations team includes:
  - Corporate Cybersecurity
  - Cybersecurity Operations Center (CSOC)
  - Detection and Response Team (DART)
  - Cybersecurity Threat Intelligence (CTI)
- Our data platform is called TRAILS (Threat Response Analytics and Intelligence Lake Service)
- We built TRAILS to serve the following Cybersecurity functions:
  - Ingest and ETL to refined source tables for investigations and threat hunting
  - Standardized real-time and scheduled detections allow for automated security and cross functional alerting
  - SQL and Delta allow for large scale data analysis using joins and complex queries

Chris Mandich
Director, Cybersecurity Operations
cmandich(at)rivian.com

Rob Hartman
Staff Cybersecurity Engineer
roberthartman(at)rivian.com

WHEN YOU TELL YOUR NON-TECH FRIEND YOU WORK WITH TRAILS
AND THEY THINK YOU'RE A PARK RANGER

Our Data Journey

How did we get here?

- Our requirements for a modern security data platform were outgrowing the capabilities of our previous platforms
  - We have migrated between 3 SIEM platforms over the past three years
  - Over the past three years our data volume has increased 30x
  - The cost to renew our previous platform was increasing 2x YoY
- Motivated by our requirements and guiding principles we considered an alternative approach for our data platform

Migration to TRAILS Timeline

- **October 2023** Detection Engine POC
- **November 2023** Data source migration start
- **December 2023** Detection migration start
- **January 2024** Data source migration complete
- **January 2024** Detection migration complete

Guiding Principles for Migration

What really matters to us?

Cost Management

- Reduce cost through control of the data and infrastructure
- Eliminate volume based ingest licensing
- Align with the strategic direction of the business

Data Ownership

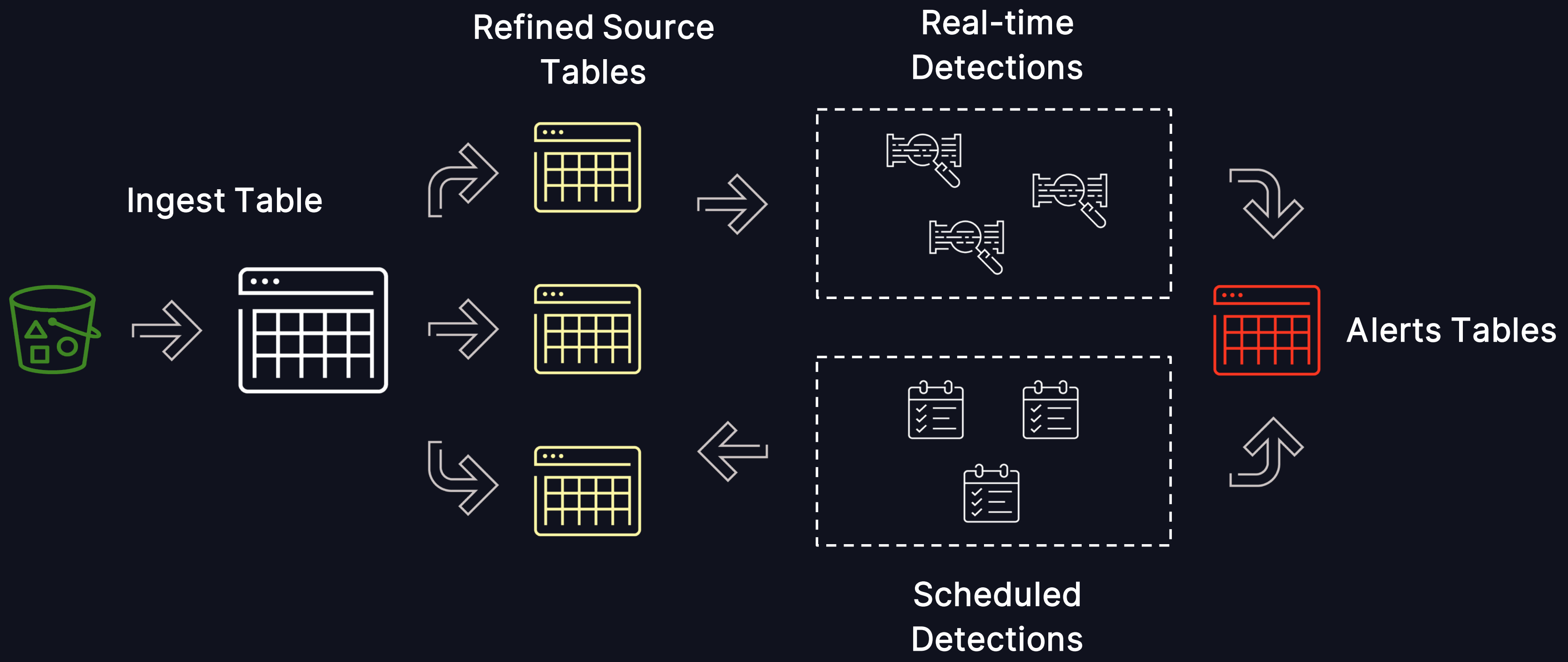
- Ability to move our data wherever we want, whenever
- Keep our data in our own environment
- Set our own SLAs on how long the data should be available

Scalability

- Ability to adjust compute and storage as needed
- Reduce operational overhead and platform management
- Ability to scale with surges in data volume

TRAILS High Level

What does it look like?

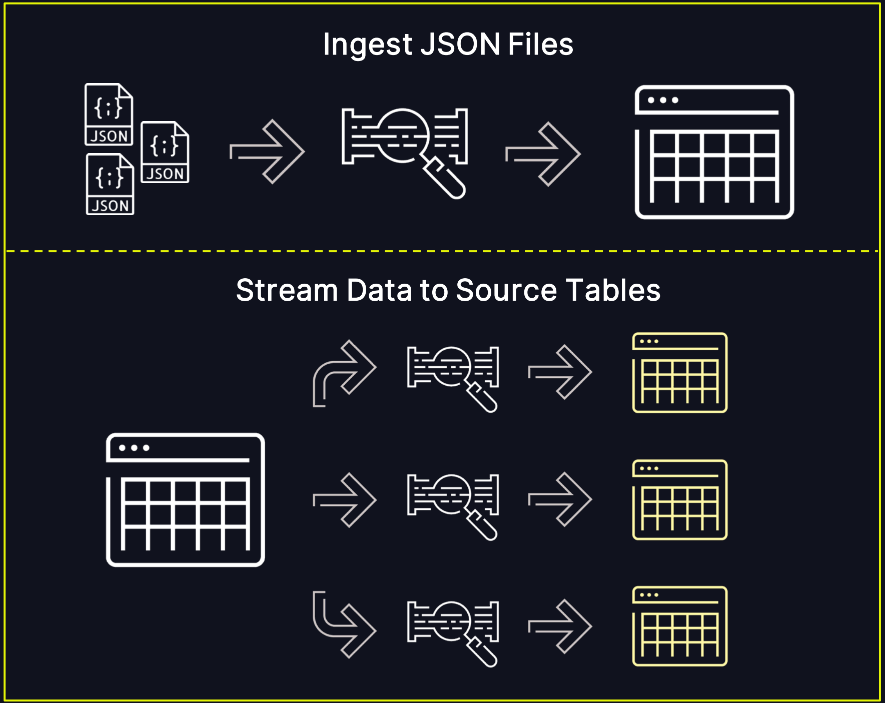


INGEST AND ETL

Foundation of TRAILS

How do we get the data?

- We ingest all data as JSON strings into one large cybersecurity_ingest Delta table
- The ingest table is partitioned by data source
- Each data source has a streaming job that populates a refined source table
- Every refined source table has the raw event stored in a _raw column for auditability
- ETL jobs remap source JSON fields to normalized data model columns



ETL and Normalization

How do we transform the data?

- During our migration of data sources, we chose to normalize the refined data source tables to ECS (Elastic Common Schema)
- We chose to use ECS based on the maturity and support in the cybersecurity community
- All source JSON fields are remapped into columns to support detections and investigation queries
- The event_timestamp, _date, and _raw columns are all passed from the ingest table to the refined source table
- All refined source tables are partitioned by _date (hint: don't over partition)

PYTHON

```python
. . .

# use attribute.subattribute to get nested items in the key
logs_schema = {
  "_raw": "_raw",
  "_raw.src_ip": "source_ip",
  "_raw.dest_ip": "destination_ip",
  "_raw.src_port": "source_port",
  "_raw.dest_port": "destination_port",
}

# extract the json field values from the ingest value
schema_for_streaming = []
for field, remapping in logs_schema.items():
  column_name = remapping
  schema_for_streaming.append(column_name)
  logs_raw = logs_raw.withColumn(column_name, get_json_object(col("value"),
f"$.{field}"))

# stream the parsed data into the refined source table
df_extracted = logs_raw.select(
  col("event_timestamp").alias("ts"),
  "_date",
  *[col(column_name).alias(column_name) for column_name in
schema_for_streaming]
).writeStream.option("checkpointLocation",
checkpoint_location.toTable(f"{catalog}.{database}.{destination_table}")
```

Incident Response and Threat Intel

How do we find the needle in the haystack?

- Our investigation search times decreased by orders of magnitude using date partitioning and Delta columnar stores
- We introduced Python notebooks that allow our team to do “select * from *” searches
- We can programmatically and dynamically search data in TRAILS to provide additional context to our security alerts

PYTHON

```python
# populate the value that you want to search
values_to_lookup = ["1.1.1.1", "2.2.2.2"]
time_range = "15 days" # use the interval syntax here

# enter the column names that you want to search against in TRAILS
# example: ["source_ip", "destination_ip"]
searching_for_col = ["source_ip"]

# enter the table family or tables that you want to search against, example
["data_source_1", "data_source"]
# leave an empty list if you want to search all tables, example []
table_families = ["data_source_..."]

# if you want to include all the other columns in the table in the results
include_other_cols = False

# run this cell to get the results
tables_with_indicator_col, value_to_lookup_sql = tables_with_indicators()
spark_df, source_counts = get_results(tables_with_indicator_col,
value_to_lookup_sql)
plot_results(source_counts)

# loads the results into a temp view and can be queried below
spark_df.createOrReplaceGlobalTempView("cyber_results_tmp_view")
```

DETECTION ENGINEERING

Detection as Code

How do we automate the deployment of detections?

- The primary goal is to detect threats as early as possible, reducing the time attackers have inside a network or system
- With our X TEAM, cross functional, alerts we can leverage subject matter experts when we need more context

Detection Development

- Detection engineers write all detections in SQL
- SQL workspace is used to craft the logic
- A centralized repo is used to manage all detections

Detection Testing

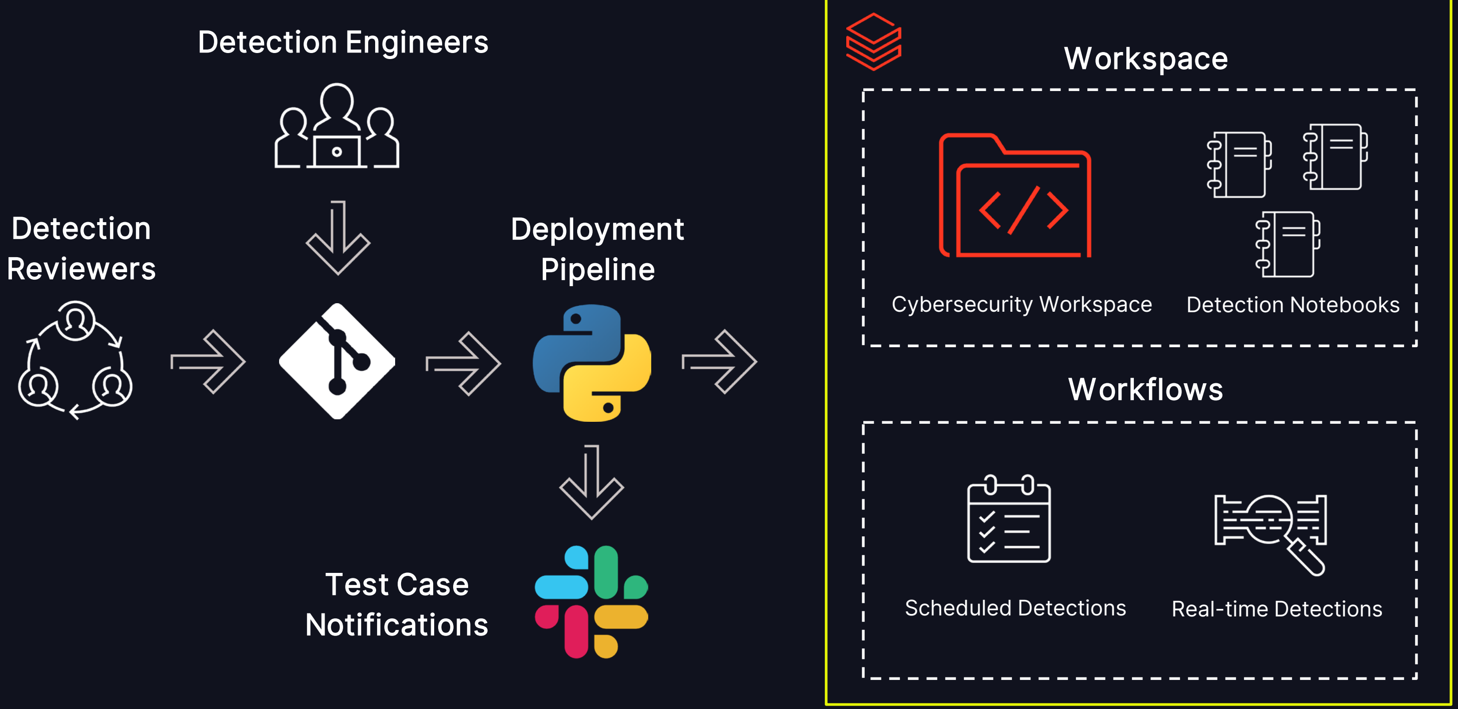
- Template attributes are validated
- Test SQL queries are run in the deployment pipeline for detection logic and column name verification
- Detections can be set into a dev state before production

Detection Deployment

- Deployed on independent job compute clusters
- Can be deployed as a scheduled or continuous workflows
- Failures are monitored through job notification webhooks and Slack

Detection Engineering Pipeline

What does the process look like?



Detection Template

What does the template look like?

```
enabled: true
name: DETECT - New user added to environment
detailed_description: |
  Detect when a new user is added to the environment.
id: a3ee860e-7c26-4316-9fcf-c9d291b9d31b
version: 1
date: '2024-06-09'
author: user@rivian.com
title: DETECT - New user (col(user_name)) added to the environment - col(ucf_urgency)
description: A new user was created and given (col(host_name)) as a computer.
source_table: refined_source_table
identity_table_link: user_name
asset_table_link: host_name
sql_search: >
  where event_action = "create account"
ucf_urgency: {'priority': 'col(identity_priority)', 'severity': 'low'}
deployment:
  action:
    suppress:
      fields:
        - user_name
      period: 1 day
  alerts_webhook:
    enabled: true
    playbook: New User Added
  prod: true
  compute:
    cluster_size: "rd-fleet.xlarge"
    max_workers: 0
metrics:
  observable:
    - ip:
      - source_ip
```

```
scheduling:
  job_run_schedule: "0 */30 * ? * *"
  earliest: 4 hours
```

Key Attributes

Title and description: can be used to make variable text that uses column/row values when wrapped in col()

Source table: the source table to stream or read from

Identity table link: the column on the source table to join on the identity lookup table

Asset table link: the column on the source table to join on the asset lookup table

SQL search: the SQL logic to use in the detection when the query runs

Suppress: suppression column values are used to make a deduplication value and suppress alerts for the period specified in the period attribute

Alerts webhook: whether to send the alert downstream

Compute: the size and max workers to use for the job compute

Observable: the columns to pull the values from to send downstream

Scheduling: for scheduled detections, the frequency and amount of time to look back in the data

Alert Tables

What do the alerts look like?

- The alert title and description include values from the rows and columns in the refined source table
  - user_name is added into the title from the evens that triggered the alert
  - ucf_urgency is added the title from the identity_priority from the joined identity lookup table and severity in the detection template
  - host_name is added to the description
- The query (alert_query) included in the alert can be executed in Databricks against the refined source table to view the detailed events that triggered the detection
- Observable values from the source_ip column are constructed

alert_title	alert_description	alert_query	observables
DETECT – New user (**testuser**) added to the environment - low	A new user was created and given (**cmp1**) as a computer.	SELECT * except (_raw) FROM source_table WHERE **event_action = "create account"** AND ts <= '**2024-06-09 23:51:31.084**' AND ts >= '**2024-06-09 23:45:20.585859**' AND user_name LIKE '%**testuser**%' AND _date >= '**2024-06-09**' AND _date <= '**2024-06-09**' ORDER BY ts desc	{"ip":["**1.1.1.1**"]}
DETECT – New user (**baduser**) added to the environment - low	A new user was created and given (**cmp19**) as a computer.	SELECT * except (_raw) from source_table where **event_action = "create account"** AND ts <= '**2024-06-09 23:51:31.987**' AND ts >= '**2024-06-09 23:45:20.585859**' AND user_name LIKE '%**baduser**%' AND _date >= '**2024-06-09**' AND _date <= '**2024-06-09**' ORDER BY ts desc	{"ip":["**1.1.1.2**"]}
DETECT – New user (**gooduser**) added to the environment - low	A new user was created and given (**cmp42**) as a computer.	SELECT * except (_raw) FROM source_table WHERE **event_action = "create account"** AND ts <= '**2024-06-09 19:14:18.293**' AND ts >= '**2024-06-09 23:45:20.585859**' AND user_name LIKE '%**gooduser**%' AND _date >= '**2024-06-09**' AND _date <= '**2024-06-09**' ORDER BY ts desc	{"ip":["**1.1.1.3**"]}

RESPONSE

Guiding Principles for Response

How do we respond to all the alerts?

Scalability

- Architecture should support scaling to accommodate growth in alert volume
- Provide a modular platform that can expand with future use cases

Context

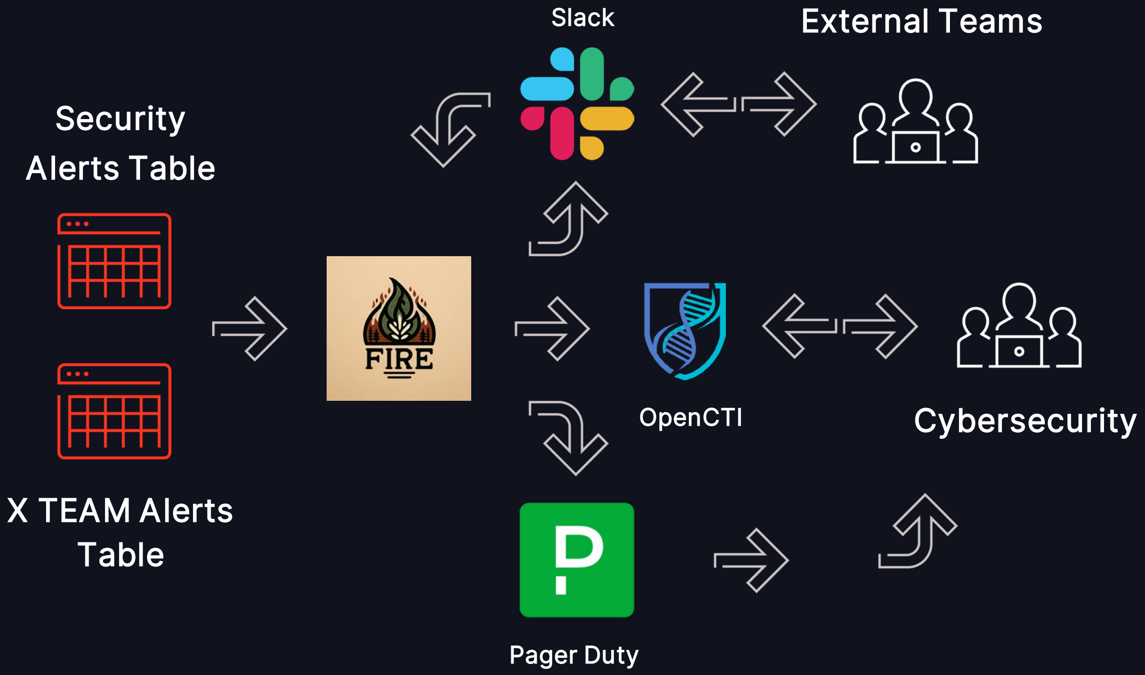
- Monitor for trends in the environment
- Provide a holistic picture of our detection coverage
- Ability to have external team SMEs provide in-depth context for an alert

Reliability

- Respond to alerts prioritized by urgency
- Ensure high availability and fault tolerance to minimize downtime

Alert Automation

Where do all the alerts go?



FIRE

(Front-line Incident Response Engine)

FIRE was built as a modular alert automation and orchestration platform.

FIRE currently supports the following integrations:

- Slack for alert notifications and interactions
- Pager Duty for security escalations and external team incidents
- OpenCTI for case management using STIX (Structured Threat Information eXpression) objects to create alerts, cases, and observables

The intention of FIRE is interchangeability, built with an event driven architecture in AWS with SQS and Lambda functions.

The integrations library can be expanded or replaced at any time.

WHAT'S NEXT FOR RIVIAN?

Continuously Building

How can we expand this platform?

- Dashboarding for common security operations and incident response use cases
- Integrate a language model to help facilitate threat hunting and incident response
- Deploy self-service features for data onboarding and normalization
- Expand our open-source footprint in the environment, from data collection to case management

DATA+AI SUMMIT