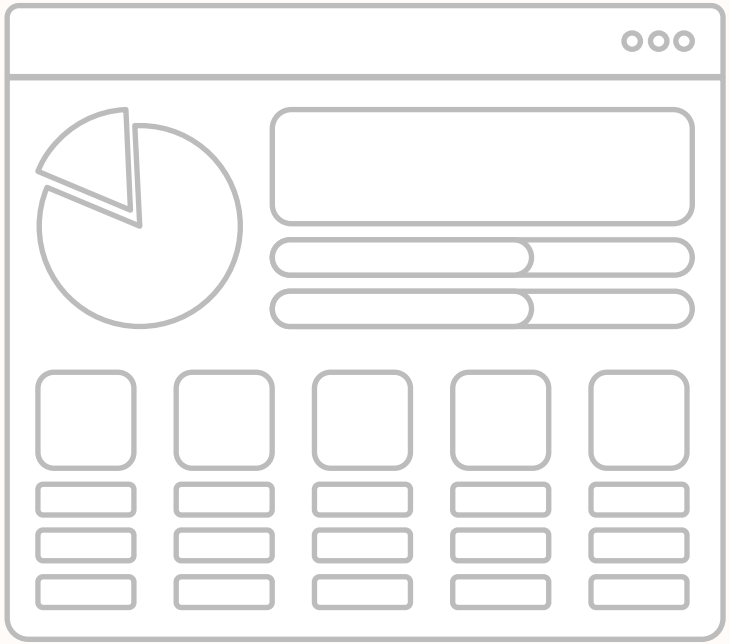
CANADIAN CENTRE FOR CYBER SECURITY

Streaming Sigma Detection at Scale

Communications Security Establishment
Centre de la sécurité des télécommunications
Canada

Table of contents



CANADIAN CENTRE FOR CYBER SECURITY (CCCS)

- CCCS believes that security is a team sport – we all must work together to be successful
- We strive to be transparent and share tools and techniques with partners
- We make it a priority to contribute to and develop open-source tools
- We share information via blog posts and online forums to allow for open-communication and relationship building

DATA FLOW OVERVIEW

JSON

MsgPack

CEF

Data Normalisations

- Parse/Decode
- MA Labelling
- Enrichment

Streaming Analytics

Streaming

Batching

Data Engineering

- Base tables
- Fact tables
- Dimension tables

Batch Analytics

Analytics Output

PROJECT OBJECTIVES

01

Simplify analytic creation with Sigma

02

Reduce time to action compromises

03

Improve scalability

04

Reduce costs of storage and compute

RESEARCH PROJECT - UPDATE

We built a proof of concept

We are now transitioning into production

Productions metrics will be released soon!

KEY TECHNOLOGIES

ICEBERG

STORAGE AND QUEUE

SIGMA

DETECTION LOGIC

STREAMING ENGINE

Spark Streaming

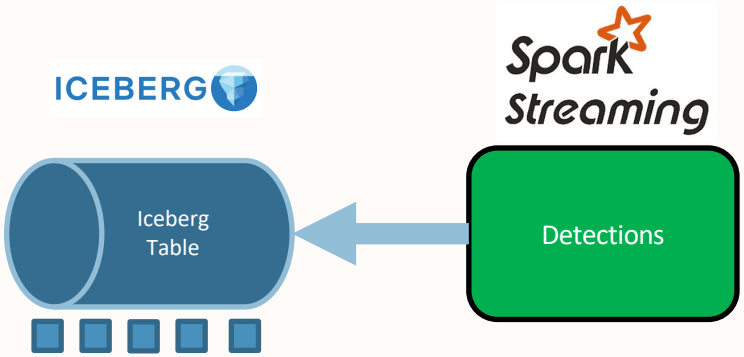
ICEBERG TABLE

```
└── ip_flow
    ├── metadata
    │   └── snapshot.avro
    └── data
        ├── day=344
        │   ├── ...
        │   │
        │   ├── hour=11
        │   │   └── data.parquet
        │   ├── hour=12
        │   │   └── data.parquet
        │   └── ...
        └── day=345
            ├── ...
            │
            ├── hour=05
            │   └── data.parquet
            ├── hour=06
            │   └── data.parquet
            └── ...
```

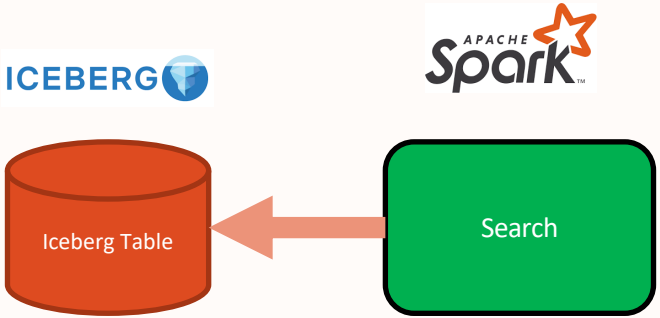
- A table format developed and open sourced by Netflix. Equivalent to Databricks' Deltalake.
- Only relies on a regular file system (NFS, S3, ABFS)

ICEBERG

ICEBERG TABLE HAS TWO ACCESS PATTERNS



Find what's new
using an offset



SELECT * FROM **catalog.prod.event_logs**
WHERE cmd like "notepad"

It's a low-cost solution to **store** data AND to **stream** data

ICEBERG AN ALTERNATIVE TO KAFKA

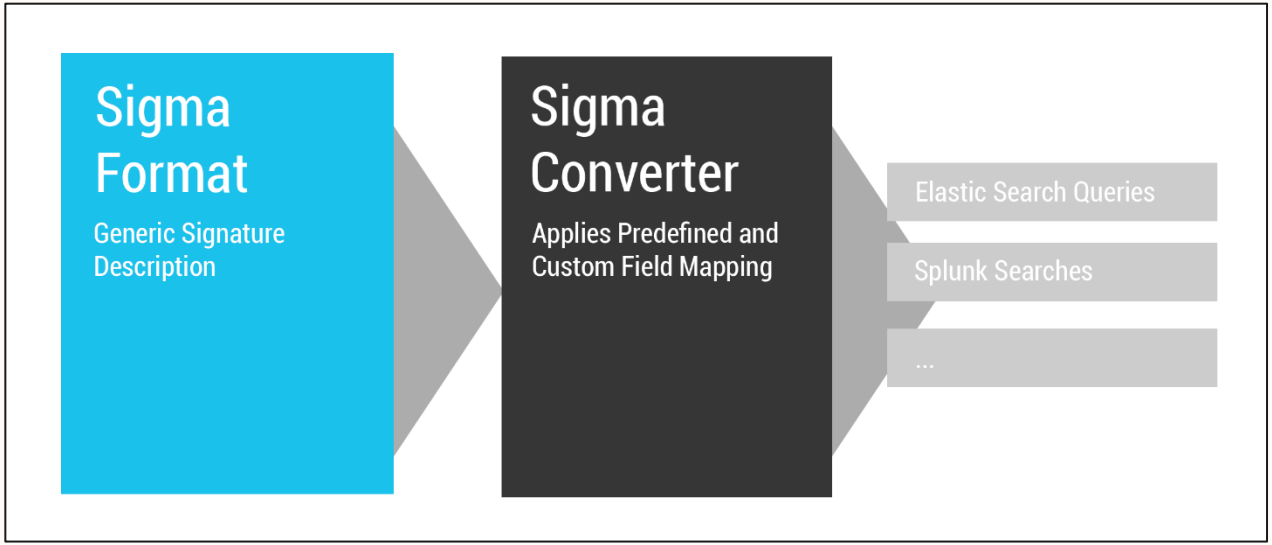
ICEBERG

1. Messaging service like Azure EventHubs are expensive
2. Operating your own Kafka cluster requires compute and staff
3. Message size can be larger than 1MB
4. Higher latency then Kafka; 10s vs 10ms
5. Upcoming Qcon talk from Apple - Near real-time and low-cost

STANDARDIZE RULE DEFINITIONS

Sigma with a streaming twist

Sigma is for log files what Snort is for network traffic and YARA is for files.



BENEFITS OF SIGMA

Open standard for detections

Portable across runtimes (ElasticSearch, SQL, QRadar, SIEM)

Common vocabulary promotes sharing

EMBRACING SQL

We are extending the existing pySigma SQL backend

trino

ORACLE

APACHE Spark™

Azure Log Analytics

EXAMPLE SIGMAHQ RULE

```yaml
title: Webshell ReGeorg Detection Via Web Logs
id: 2ea44a60-cfda-11ea-87d0-0242ac130003
status: test
description: >
    Certain strings in the uri_query field when combined with
    null referer and null user agent can indicate activity
    associated with the webshell ReGeorg.
references:
    - https://community.rsa.com/community/products/netwitness/blog/2019/02/19/
    - https://github.com/sensepost/reGeorg
author: Cian Heasley
date: 2020/08/04
modified: 2023/01/02
tags:
    - attack.persistence
    - attack.t1505.003
logsource:
    category: webserver
detection:
    selection:
        cs-uri-query|contains:
            - 'cmd=read'
            - 'connect&target'
            - 'cmd=connect'
            - 'cmd=disconnect'
            - 'cmd=forward'
    filter:
        cs-referer: null
        cs-user-agent: null
        cs-method: POST
    condition: selection and filter
falsepositives:
    - Web applications that use the same URL parameters as ReGeorg
fields:
```

```sql
%%sparksql
SELECT
    *,
    (selection AND filter) AS webshell_regeorg
FROM
    (
        SELECT
            *,
            (
                    `cs-uri-query` LIKE '%cmd=read%'
                OR `cs-uri-query` LIKE '%connect&target%'
                OR `cs-uri-query` LIKE '%cmd=connect%'
                OR `cs-uri-query` LIKE '%cmd=disconnect%'
                OR `cs-uri-query` LIKE '%cmd=forward%'
            ) AS selection,
            (
                `cs-referer` IS NULL
                AND `cs-user-agent` IS NULL
                AND `cs-method` = 'POST'
            ) as filter
        FROM
            webserver_logs_table
    )
WHERE
    webshell_regeorg
```

https://github.com/SigmaHQ/sigma/tree/master/rules

MODERN SQL AND TABLES

```
%%sparksql
select
    named_struct('name', 'Bob', 'age', 32) as user,
    array(1,2,3) as login_array,
    map('cve-2019-1214', 55, 'cve-2017-1536', 87) as cve_map

```

user	login_array	cve_map
Row(name='Bob', age=32)	[1, 2, 3]	{'cve-2017-1536': 87, 'cve-2019-1214': 55}

Results: a table with one row, but with complex columns

- No longer limited to scalar values
- Support for Struct, Array, Map
- Equivalent to working with JSON rather than CSV files

RUNNING MULTIPLE RULES IN SINGLE PASS

cs-uri-query	cs-referer	cs-method	cs-user-agent
https://a.com	https://ref	GET	mozilla
https://b.com	https://x.org	POST	edge
https://c.com	https://z.ca	GET	chrome

```
%%sparksql --jinja --output text
select
    *,
    -- regroup the rule in a map
    -- ruleName -> evaluated boolean SQL expression
    map(
        'Webshell ReoGeorg Detection Via Web Logs',
        {{generated_rule1_expression}}
        ,
        'Suspicious URL parameters',
        {{generated_rule2_expression}}
        ,
        'Suspicious User-Agents Related To Recon Tools',
        {{generated_rule3_expression}}

    ) as sigma_tests
from
    webserver_logs_table
```

cs-uri-query	cs-referer	cs-method	cs-user-agent	sigma_tests
https://a.com	https://ref	GET	mozilla	{'Webshell ReoGeorg Detection Via Web Logs': True, 'Suspicious User-Agents Related To Recon Tools': False, 'Suspicious URL parameters': False}
https://b.com	https://x.org	POST	edge	{'Webshell ReoGeorg Detection Via Web Logs': True, 'Suspicious User-Agents Related To Recon Tools': False, 'Suspicious URL parameters': False}
https://c.com	https://z.ca	GET	chrome	{'Webshell ReoGeorg Detection Via Web Logs': True, 'Suspicious User-Agents Related To Recon Tools': False, 'Suspicious URL parameters': False}

HOW TO RUN THE RULES

BATCH VS. STREAMING

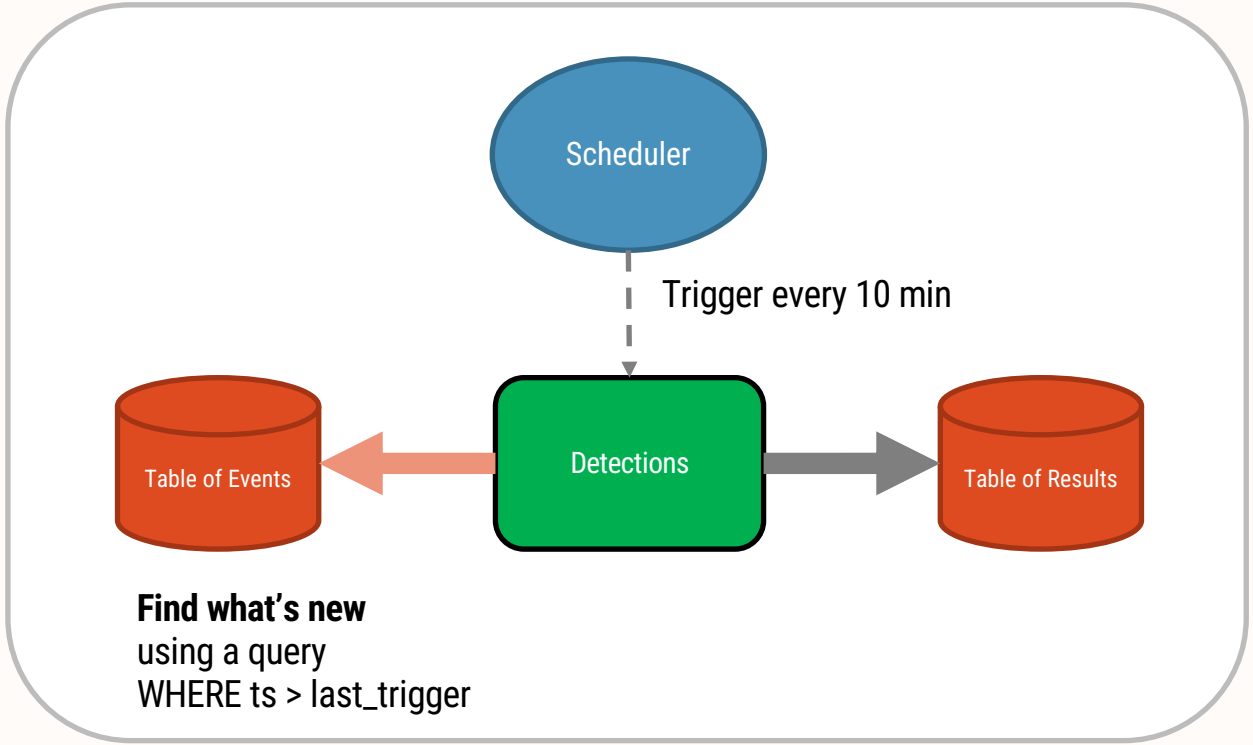
BATCH MODE

Table of Events must be ingested and indexed (not shown here)

Table is queried at every trigger

spark.**read**.format("iceberg")

Last successful batch time must be checkpointed in a fault resilient way

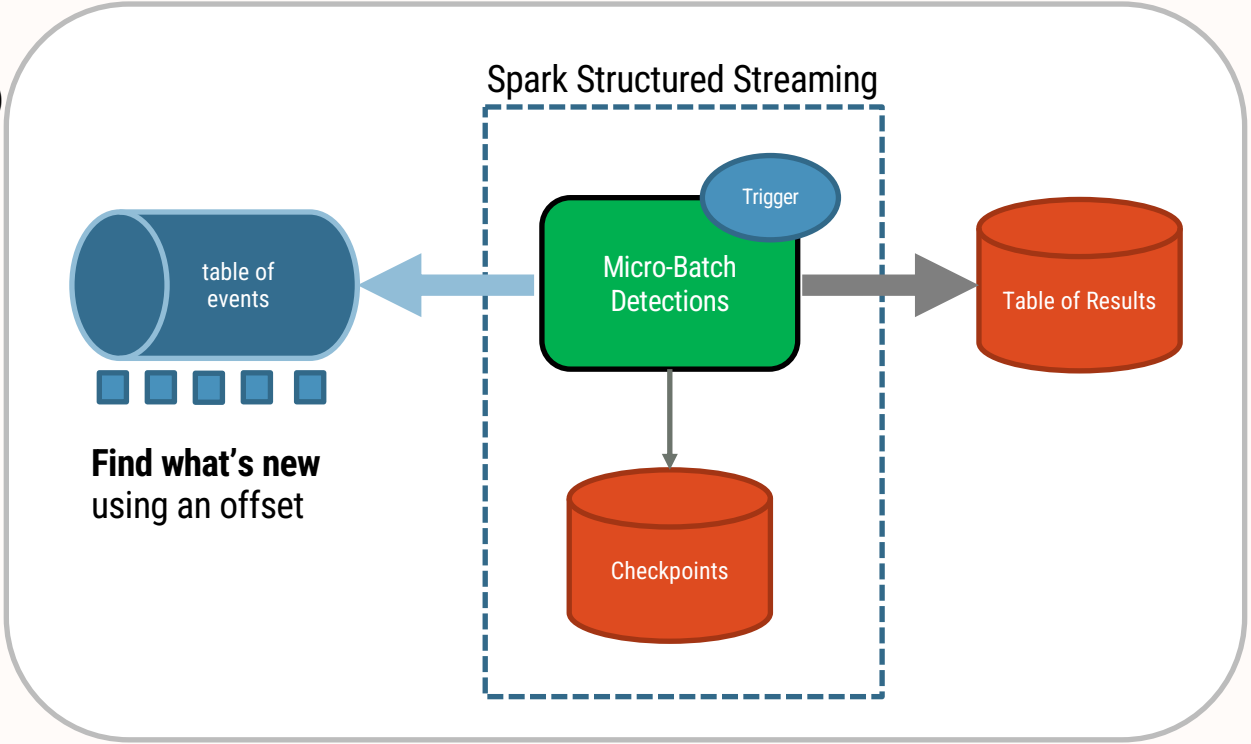


STREAMING MODE

spark.**readStream**.format("iceberg")
Instead of
spark.**read**.format("iceberg")

Spark manages checkpoints

Rich API
Spark Structured Streaming



AGGREGATIONS

Contrary to discrete detections, aggregations require **multiple** events.

For example, **count** the number of failed login attempts in the last three hours.

```yaml
title: High DNS Bytes Out
id: 0f6c1bf5-70a5-4963-aef9-aab1eefb50bd
status: unsupported
description: High DNS queries bytes amount from host per short period of time
author: Daniil Yugoslavskiy, oscd.community
date: 2019/10/24
modified: 2023/03/24
tags:
    - attack.exfiltration
    - attack.t1048.003
logsource:
    category: dns
detection:
    selection:
        query: '*'
    timeframe: 1m
    condition: selection | sum(question_length) by src_ip > 300000
falsepositives:
    - Legitimate high DNS bytes out rate to domain name which should be added to whitelist
level: medium
```

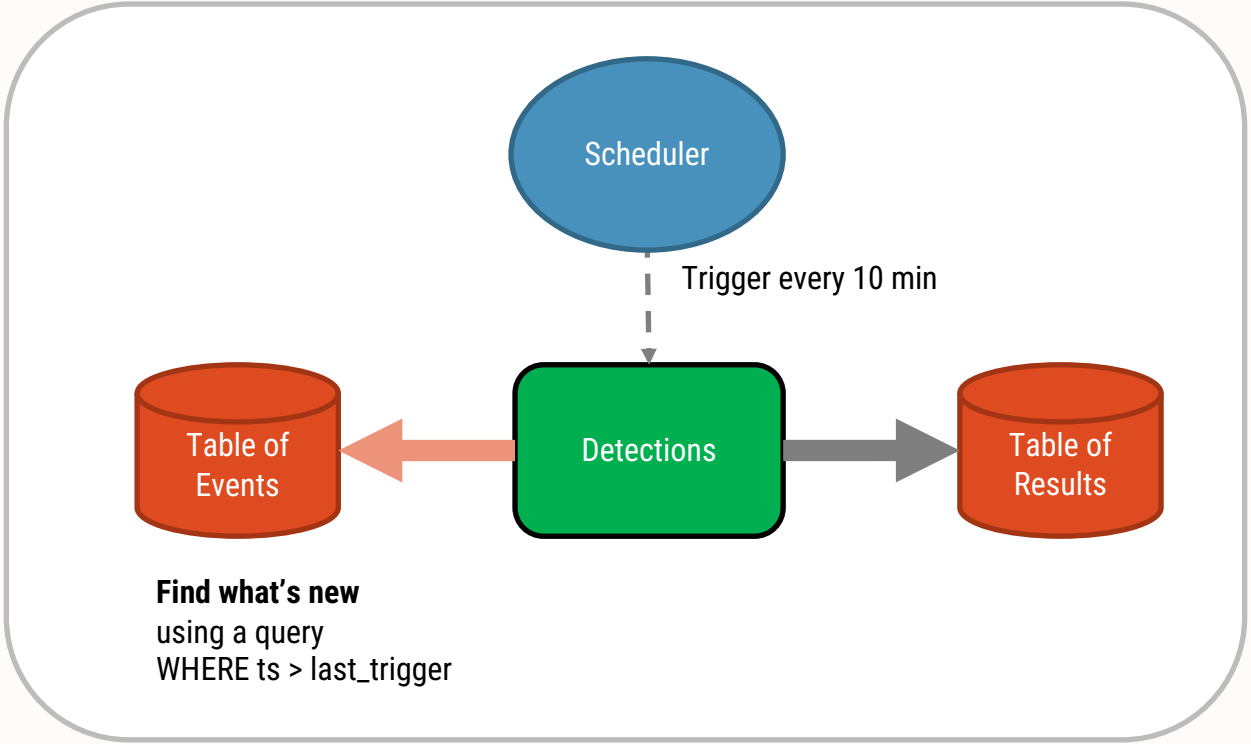
```yaml
title: Enumeration via the Global Catalog
description: Detects enumeration of the global catalog (that can be perf
status: experimental
author: Chakib Gzenayi (@Chak092), Hosni Mribah
id: 619b020f-0fd7-4f23-87db-3f51ef837a34
date: 2020/05/11
modified: 2022/08/15
references:
    - https://docs.microsoft.com/en-us/windows/security/threat-protectio
tags:
    - attack.discovery
    - attack.t1087.002
logsource:
    product: windows
    service: security
    definition: 'The advanced audit policy setting "Windows Filtering Pl
detection:
    selection:
        EventID: 5156
        DestPort:
        - 3268
        - 3269
    timeframe: 1h
    condition: selection | count() by SourceAddress > 2000
falsepositives:
    - Exclude known DCs.
level: medium
```

AGGREGATION IN BATCH MODE

Check for more than 20 login attempts in last 3 hours

Results in **re-processing** 3h worth of data every 10 min

Decreasing the trigger time, increase the cost

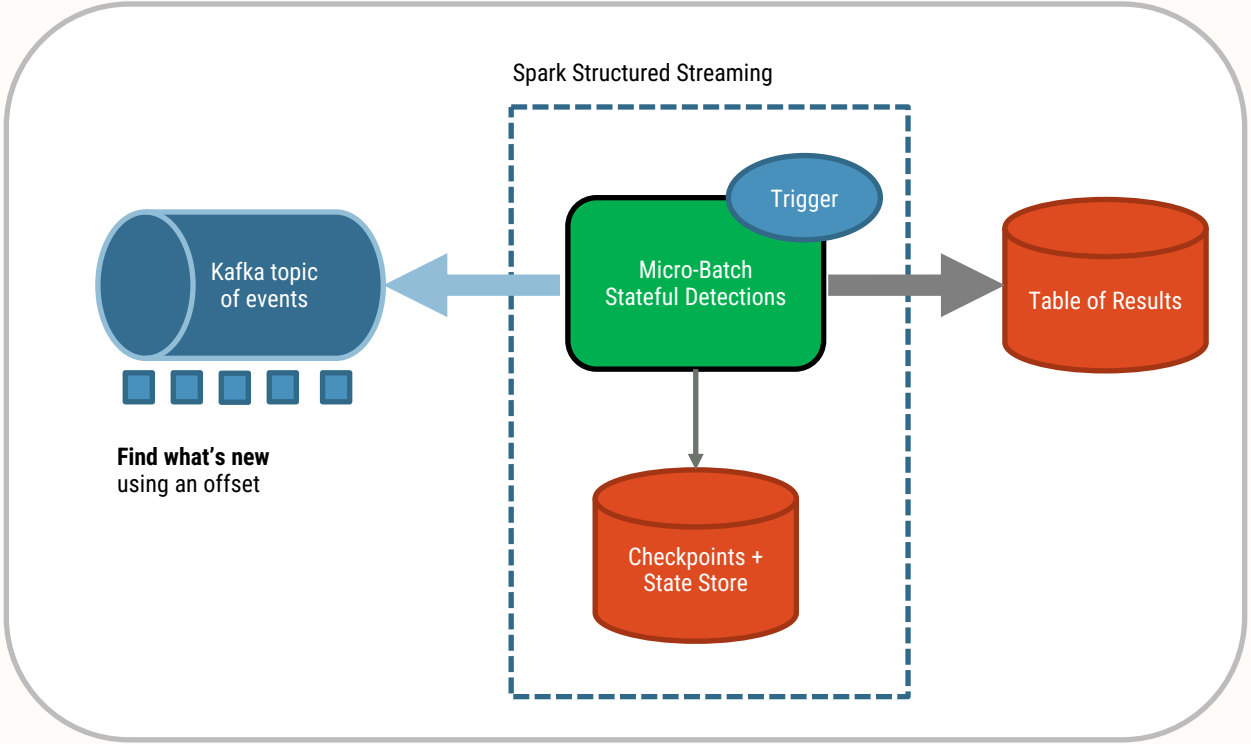


AGGREGATION IN STREAMING MODE

Spark maintains an in-memory state and takes care of persisting it

Aggregation are incrementally updated

Events are process only once



JOINs

Required when the rule refers to past attributes

Sigma rule referencing CommandLine of past event.

This assume a JOIN

```yaml
title: UAC Bypass Using Windows Media Player - Process
id: 0058b9e5-bcd7-40d4-9205-95ca5a16d7b2
status: test
description: Detects the pattern of UAC Bypass using Windows Media Player osksupport.dll (UACMe 32)
references:
    - https://github.com/hfiref0x/UACME
author: Christian Burkard (Nextron Systems)
date: 2021/08/23
modified: 2022/10/09
tags:
    - attack.defense_evasion
    - attack.privilege_escalation
    - attack.t1548.002
logsource:
    category: process_creation
    product: windows
detection:
    selection1:
        Image: 'C:\Program Files\Windows Media Player\osk.exe'
        IntegrityLevel:
            - 'High'
            - 'System'
    selection2:
        Image: 'C:\Windows\System32\cmd.exe'
        ParentCommandLine: '"C:\Windows\system32\mmc.exe" "C:\Windows\system32\eventvwr.msc" /s'
        IntegrityLevel:
            - 'High'
            - 'System'
    condition: 1 of selection*
falsepositives:
    - Unknown
level: high
```

Windows Event 4688

We have the current CommandLine

But we don’t have the ParentCommandLine

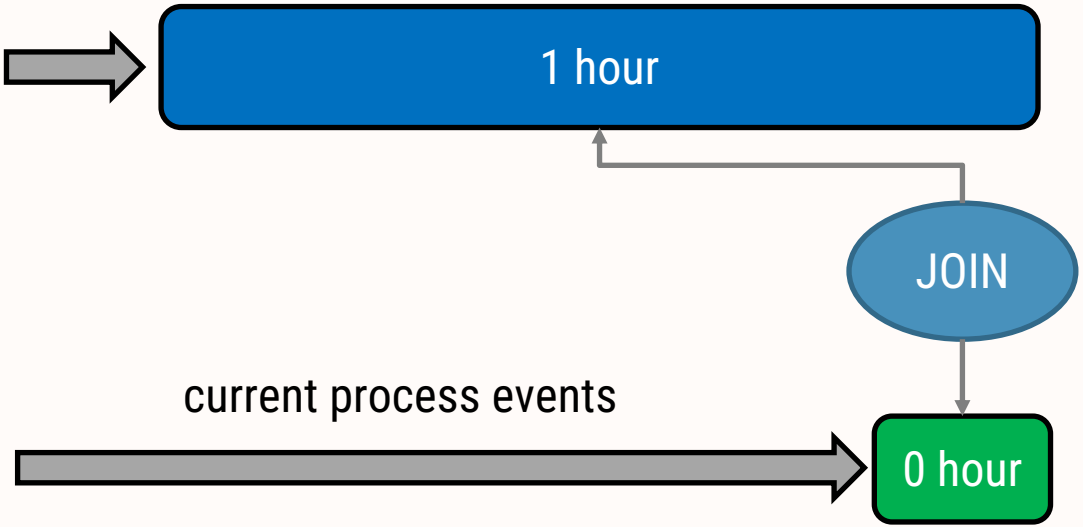
We need to JOIN with past events

Examples of 4688

Windows 2016/10

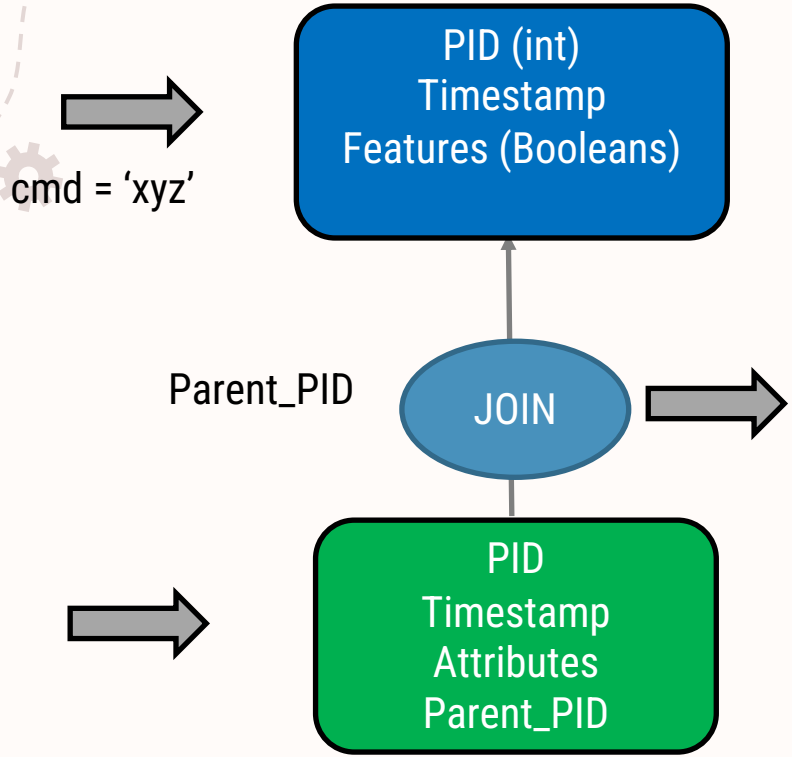
```
A new process has been created.

Creator Subject:
 Security ID:  SYSTEM
 Account Name:  RFSH$
 Account Domain:  LAB
 Logon ID:  0x3E7

Target Subject:
 Security ID:  LAB\rsmith
 Account Name:  rsmith
 Account Domain:  LAB
 Logon ID:  0x2C9D82

Process Information:
 New Process ID:  0x2e0e4
 New Process Name: C:\Windows\System32\RuntimeBroker.exe
 Token Elevation Type: %%1938
 Mandatory Label:  Mandatory Label\Medium Mandatory Level
 Creator Process ID: 0x268
 Creator Process Name: C:\Windows\System32\svchost.exe
 Process Command Line:
```

SPARK STREAM-STREAM JOINs



Looking up previous start-process events requires in memory caching

We can now apply the sigma rule condition



Each cached event uses memory

- **Filtering**: Only keep parents of interest
- **Reduce** size of each message:
  - use an integer key (hash)
  - only keep the features

Evaluation results:

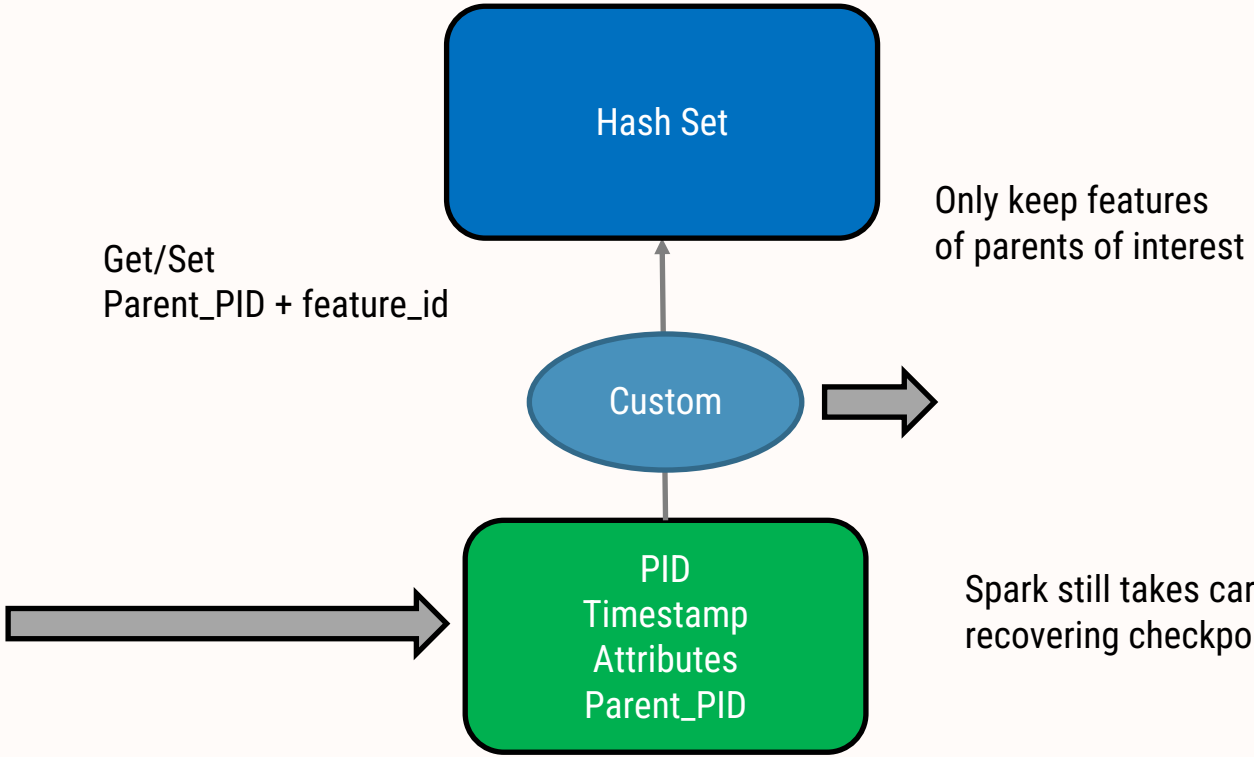
- 1 large spark worker machine
- 2,500 message/s
- Max of 160 million features

IMPROVING ON STREAM-STREAM JOINs

By reading the stream once

By reducing the memory footprint

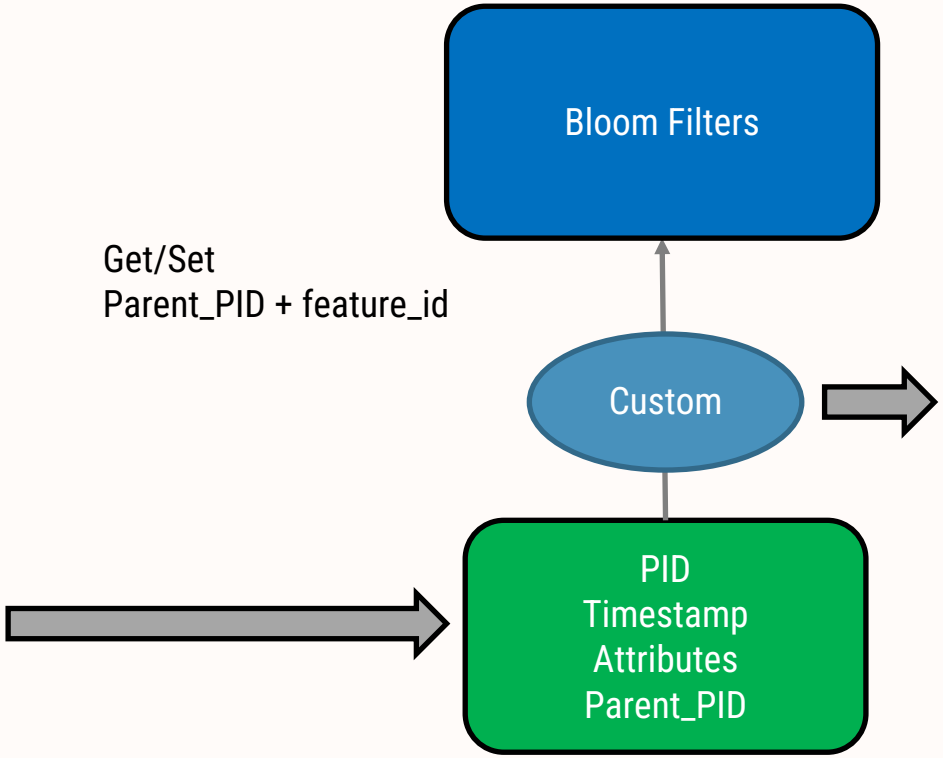
READ THE STREAM ONCE



Only keep features
of parents of interest

Spark still takes care of storing and
recovering checkpoints and state

REDUCE THE MEMORY FOOTPRINT



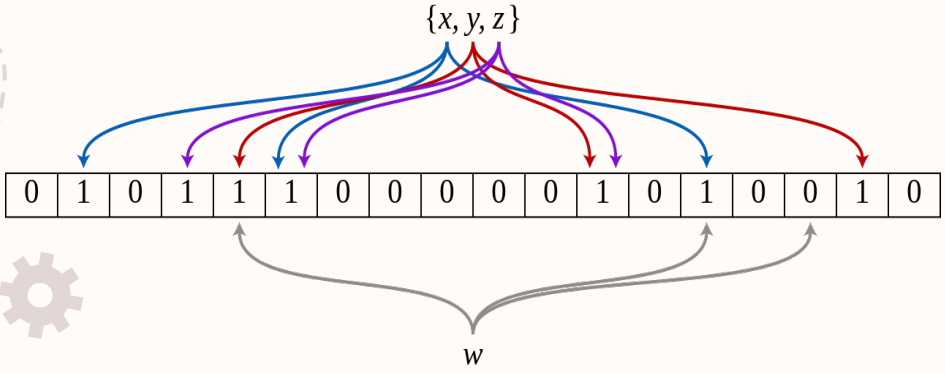
Replace Hash Set with a bloom filter

Only the **fingerprints** of keys are stored in bloom filter. It's very compact

2 orders of magnitude greater capacity

- 10,000 message/s → **4x**
- Max of 20 billion features → **100x**
- @ 0.1% false positive rate

BLOOM FILTERS



Storing; flip bits “true” in a N bit array

Random bit positions = hash(key) % N

Retrieving; false positives are possible

But false negative is not. So, we don’t miss any detections.

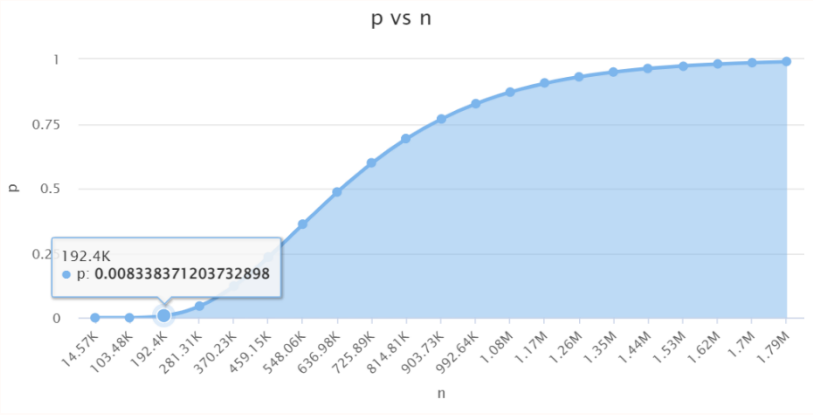
AGING OFF BLOOM FILTERS

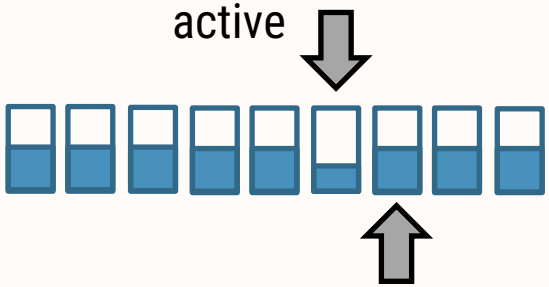
False positive probably increases over time with mathematically proven error bounds

Redis labs paper Age-Partitioned Bloom Filter

Our simple approach “ring of bloom”

https://hur.st/bloomfilter/





FLUX-CAPACITOR: Caching named observations

A simple concept that can power many use cases.

Keep track of:

- Parent
- Parent-of-parent, aka “Ancestors” (multiple JOINs)
- Temporal Proximity (upcoming SIGMA 2 specification)

✅ Read stream only once

✅ Use less memory

ANCESTORS

Telemetry

time	host	pid	ppid	child_img_match	parent_cmd_match
0	1.1.1.1	300	None	❌	✅
1	1.1.1.1	600	300	❌	❌→✅
2	1.1.1.1	900	600	✅	❌→✅

Bloom per host

Bloom_for_1.1.1.1
300.parent_cmd_match
600.parent_cmd_match

Get/Set: pid + parent_feature

TEMPORAL PROXIMITY SIGMA 2

```yaml
action: correlation
type: temporal
rule:
    - many_failed_logins
    - successful_login
group-by:
    - User
timespan: 1h
ordered: true
```

https://github.com/SigmaHQ/sigma-specification/blob/version_2/

TEMPORAL PROXIMITY

Detect presence of 3 un-ordered observations
in a time window group-by user

Telemetry

time	host	User	recon_a	recon_b	recon_c
0	1.1.1.1	Alice	❌	✅	❌
1	1.1.1.1	Alice	✅	❌→✅	❌
2	1.1.1.1	Alice	❌→✅	❌→✅	✅

Get/Set: User + tag

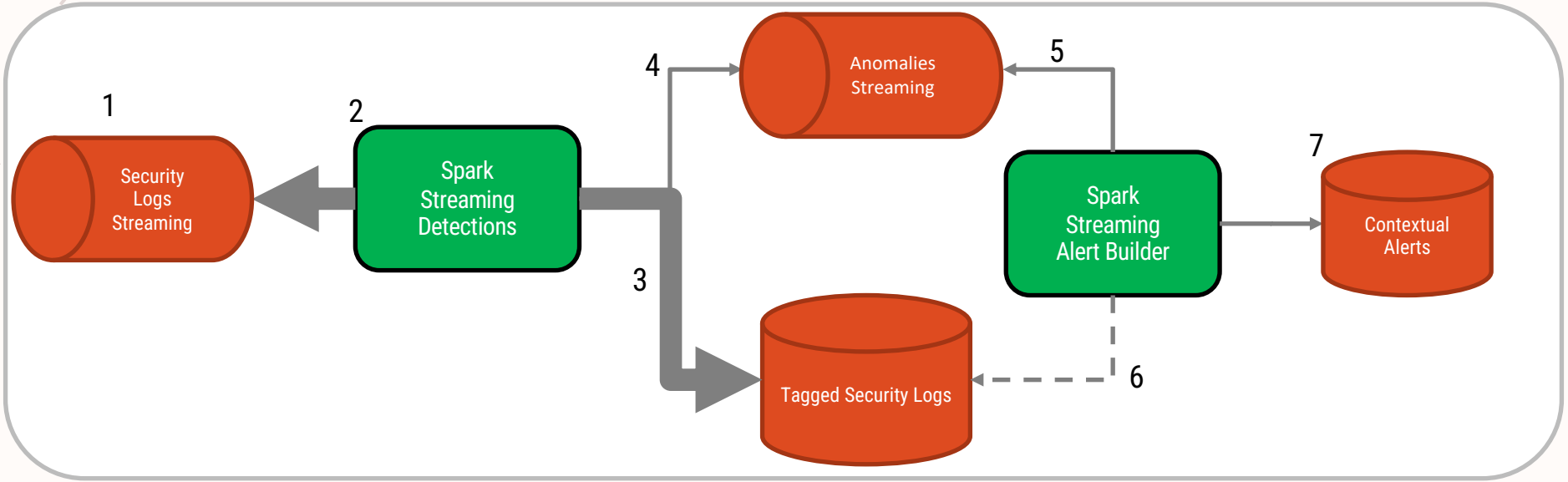
Bloom per host

Bloom_for_1.1.1.1
Alice.recon_b
Alice.recon_a

VALIDATING ANOMOLIES AND BUILDING ALERTS

Eliminating false positives

VALIDATING ANOMOLIES AND BUILDING ALERTS



Read **high** volume of fast telemetry (1)

Discrete pattern matching, handle temporality (2)

Store data partition by time and host (3)

Publish what was detected: rule name, host ID (4)

Read **low** volume of anomalies in (5)

Validate results by replaying events for each anomaly using rule name and host ID (6)

True positive results are written as alerts with pieces of evidence as a bonus (7)

EXCITING FUTURE POSSIBILITIES

Bloom filters are a form of data sketching

There are many probabilistic algorithms addressing use cases:

- Unique users
- Quantile and histogram
- Most frequent items

Lee Rhodes is the founder of Data Sketches

Open sourced by Yahoo https://datasketches.apache.org/

QUESTIONS?

Research Project Repository and Blogs

https://github.com/CybercentreCanada/flux-capacitor

https://medium.com/@jean-claude.cote

https://medium.com/@kevin.hardy-cooper