panther

Detecting Threats at Scale

Panos Sakkos - Senior Engineering Manager

Today's Agenda

Cybersecurity Ecosystem

EDR / XDR

Endpoint Detection and Response (EDR) are tools that monitor "endpoints" like desktops, laptops, mobile devices, VMs and more.

Extended Detection and Response (XDR) expands the scope of endpoints to networks, emails, cloud services and applications.

SIEM

Security Information and Event Management (SIEM) for monitoring events in real time and alerting when a suspicious event or behavior occurs.

SOAR

Security Orchestration Automation Response (SOAR) automate responses when specific alerts are triggered.

Cybersecurity Ecosystem

CSPM

Cloud Security Posture Management (CSPM) monitor cloud environments for misconfigurations and compliance violations.

CNAPP

Cloud Native Application Protection Platform (CNAPP) monitor cloud-native applications and microservices

Anatomy of a SIEM

Ingestion

Data onboarding, control and refinement.

Detections

Business logic that defines threats in order to be detected.

Security Data Lake

Storage of all events for investigation and correlation purposes.

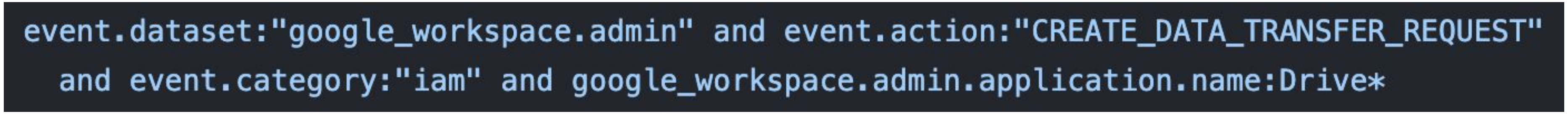
Splunk Processing Language (SPL) Detections

- Arguably the most popular language to write detections
- Not Turing Complete
- Not managed as code natively by Splunk

~~~
```
Crypto Miner User Agent

Creator: john.doe@example.test
Last updated: 2024-01-06
Rule ticket: SECURITY-1234
Documentation: internalcompanywiki.com/detection_engineering/CryptoMinerUserAgen
Source: https://github.com/SigmaHQ/sigma/blob/master/rules/web/proxy_generic/pro
```

``` ========== Filter ========== ```
_index_earliest=-30min@min _index_latest=-0min@min earliest=-24h
index=proxy (useragent="XMRig*" OR useragent="ccminer*")

``` ========== Incident properties ========== ```
| eval title = "Crypto Miner User Agent"
| eval severity = "Medium"
| eval labels = mvappend("rule:crypto_miner_user_agent","source:proxy","attack.c
| eval description = mvappend(
    "**Suspicious user agent string used by crypto miners was detected in proxy 
    " ",
    "Timestamp: ".strftime(log_time,"%F %T"),
    "Source: ".src_ip,
    "Raw event: ",
    "```"._raw."```"
    )

``` ========== Output ========== ```
| table title, severity, labels, description
~~~

Event Query Language (EQL)

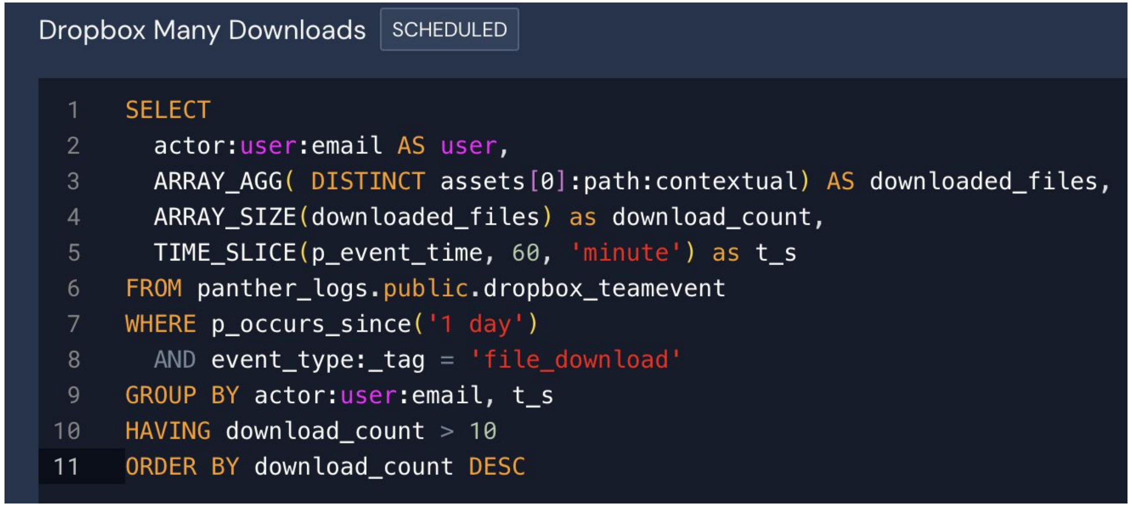
- Query language for writing Detection on Elastic Security
- Not Turing Complete
- Detections can be managed as code



```
event.dataset:"google_workspace.admin" and event.action:"CREATE_DATA_TRANSFER_REQUEST"
  and event.category:"iam" and google_workspace.admin.application.name:Drive*
```

SQL Detections

- SQL queries that run on a scheduled basis against a Data Warehouse, like Snowflake
- They trigger an alert of any results are returned
- Can be managed as code

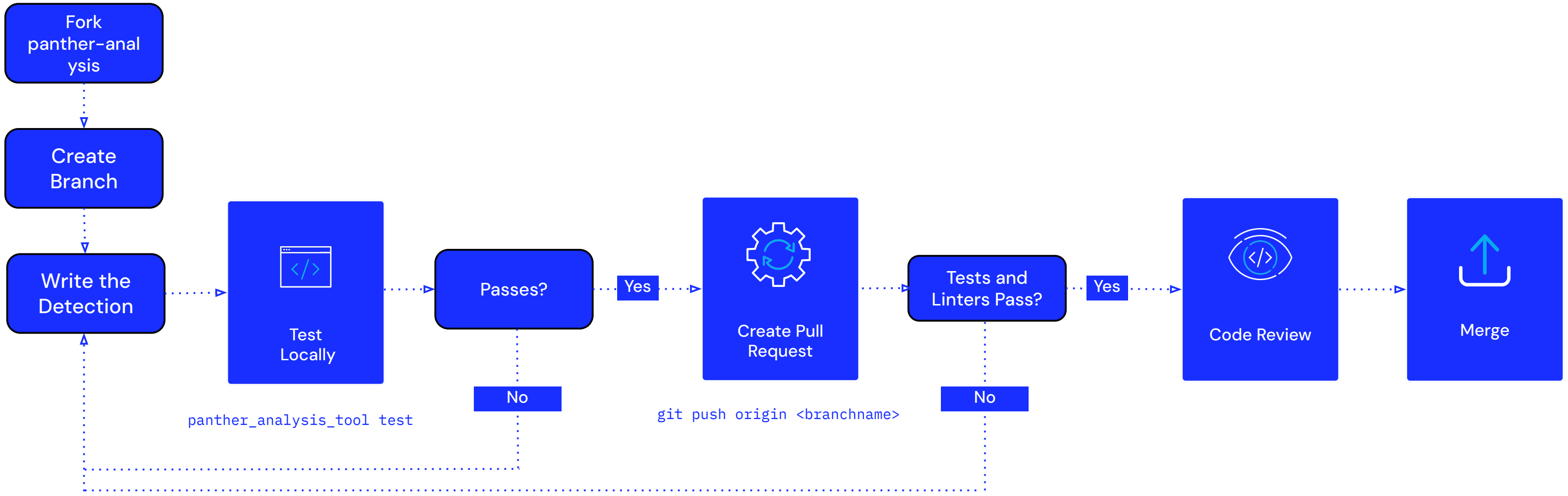


```sql
SELECT
  actor:user:email AS user,
  ARRAY_AGG( DISTINCT assets[0]:path:contextual) AS downloaded_files,
  ARRAY_SIZE(downloaded_files) as download_count,
  TIME_SLICE(p_event_time, 60, 'minute') as t_s
FROM panther_logs.public.dropbox_teamevent
WHERE p_occurs_since('1 day')
  AND event_type:_tag = 'file_download'
GROUP BY actor:user:email, t_s
HAVING download_count > 10
ORDER BY download_count DESC
```

Challenges with traditional detections

- Who changed detection X and what changed?
- When will we have detection for X use-case completed?
- Is the most current version of my logic running in production?
- Can we identify “alert storms” before they are deployed?
- If a zero-day happens, how quickly can we deploy new detections?
- Can we reuse the business logic from detection X in detection Y?

Detections as Code (DaC)



Panther Analysis repo: https://github.com/panther-labs/panther-analysis

Python Detections

- Python function that expects an event in the form of JSON and triggers and alert if it returns true
- Alerts can easily be deduplicated by the generated alert title
- Alert context can be generated in order to power automations on top of the alerts
- Custom Python libraries can be used

def rule(event):
    return True
	Returns boolean to match the log event and trigger an alert.	Required

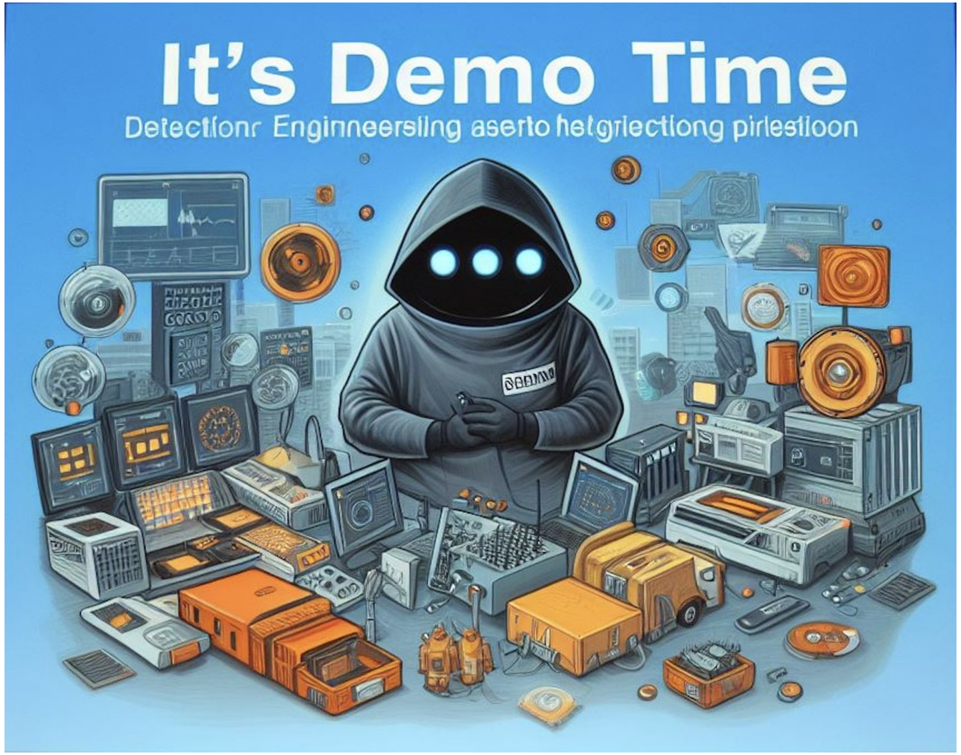
def title(event):
    return STRING
	Return a string which will be shown as the alert title.	Optional

def dedup(event):
    return STRING
	Return a string which will be used to deduplicate similar alerts.	Optional

def alert_context(event):
    return {'key':'value'}
	Return a dictionary with additional data to be included in the alert.	Optional

deep_get(event, “element”, “child element”)
    return STRING
	Returns a string for the value of a nested JSON element.	Helper function

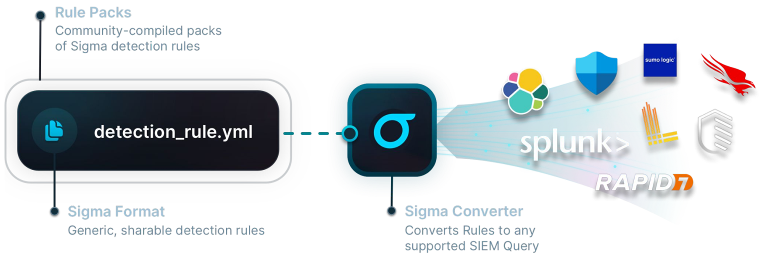
DEMO


It's Demo Time
Detectionr Enginneersiing aserto helgriectiong pirlestioon

Sigma Detections

Sigma Detections

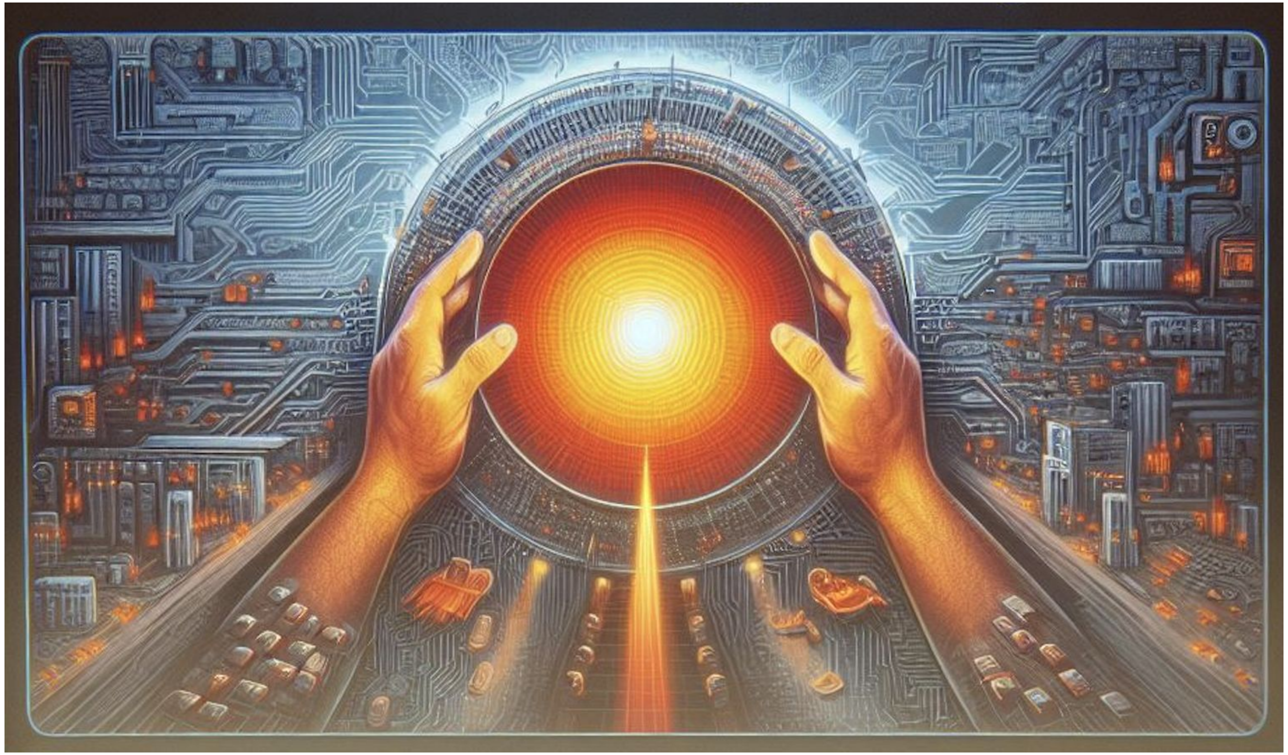
- Detections in YAML format, which can be transpiled to a variety of SIEM Detection formats
- The Sigma repository contains ~3,000 detections that are maintained by security specialists


Rule Packs
Community-compiled packs of Sigma detection rules
detection_rule.yml
Sigma Format
Generic, sharable detection rules
Sigma Converter
Converts Rules to any supported SIEM Query
sumo logic
splunk>
RAPID7

Sigma: https://github.com/SigmaHQ/sigma

Panther transpiler: https://github.com/panther-labs/pySigma-backend-panther

DEMO



panther

THANK YOU