



**Opulent Alerting:
Enriching our Lives**



Paul Harrison

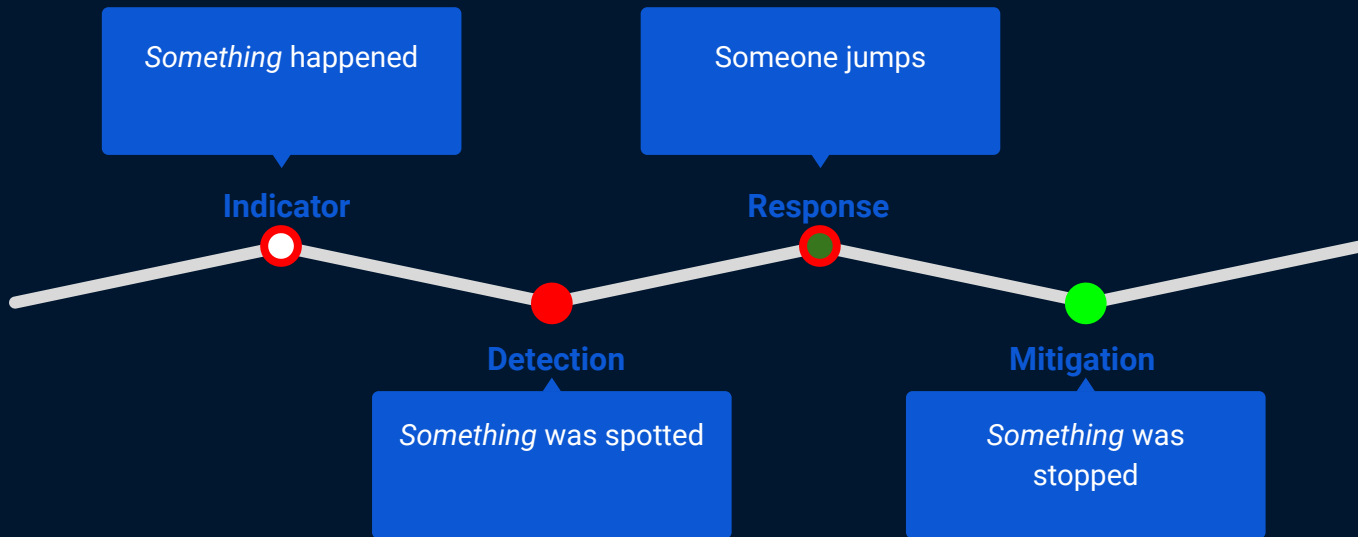
Security Operations at Mattermost

Previously

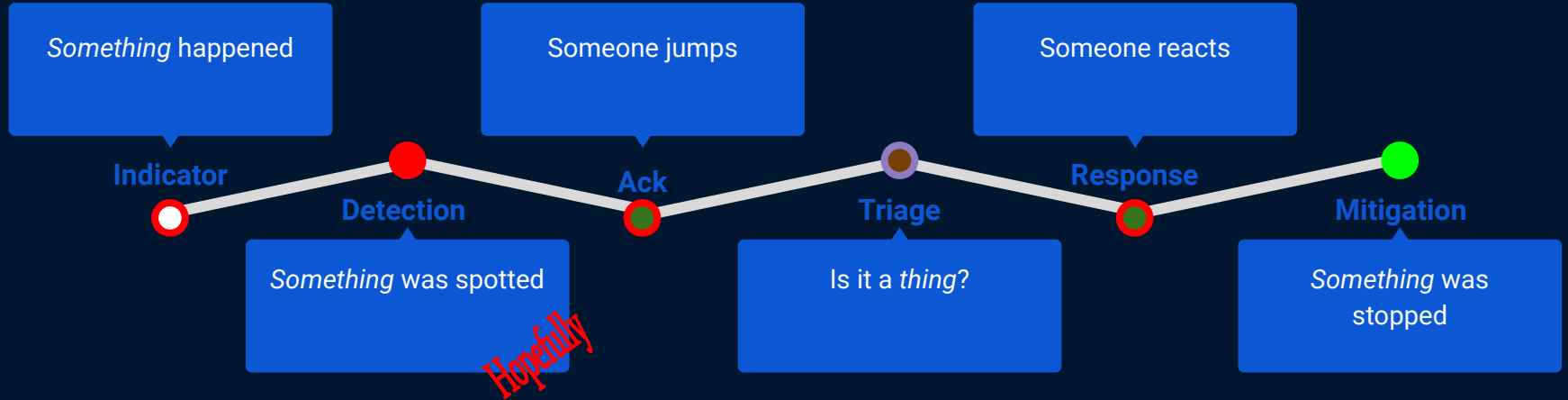
- Very Good Security
- GitLab
- Bold Commerce
- Northfield IT
- Microsoft
- Frontbridge

I build and run security operations programs

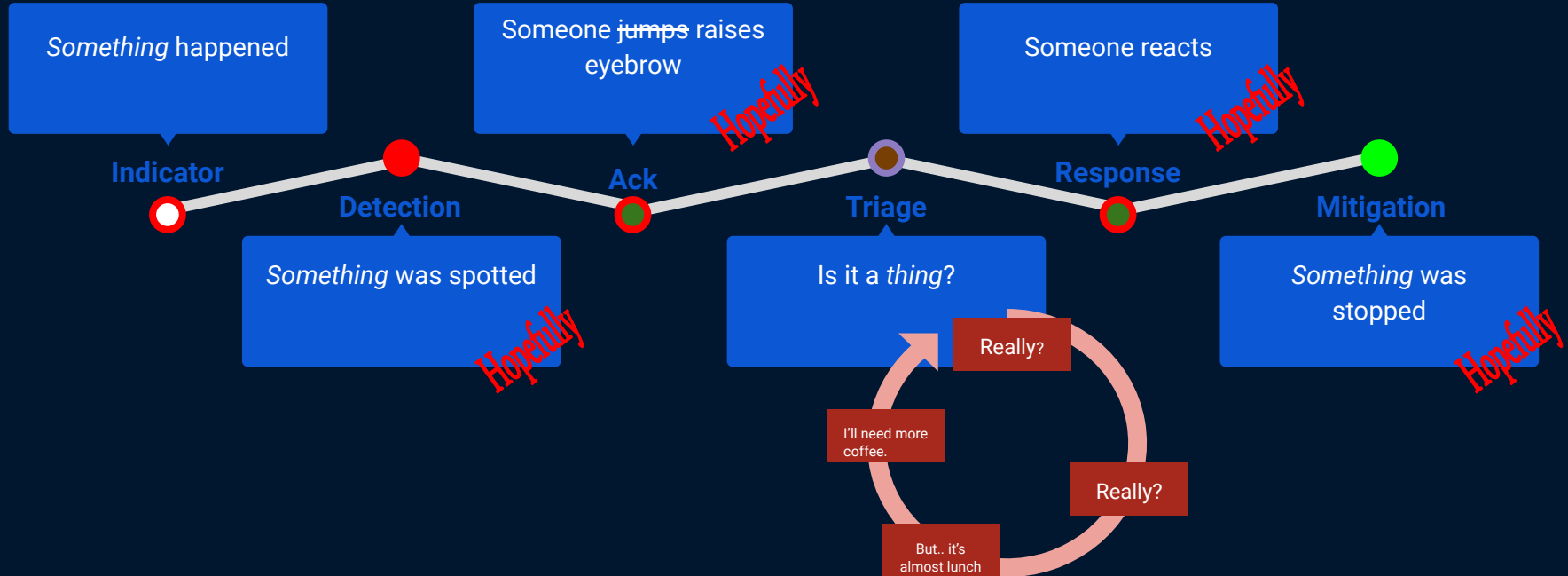
Chapter 0: Detection & Response, A Recap



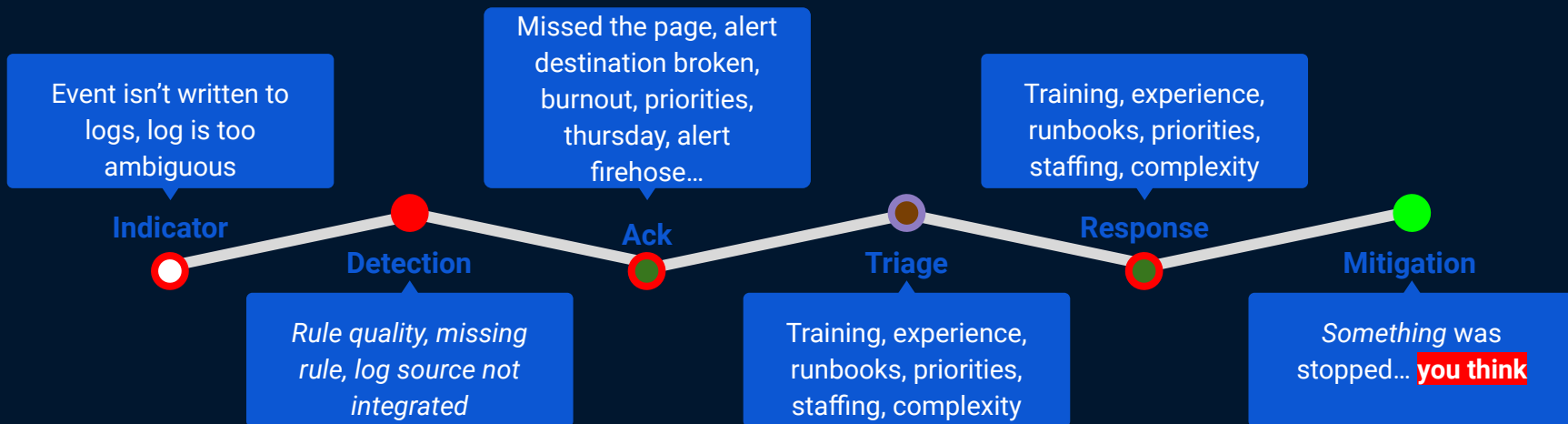
In reality...



In reality... it's a little more involved



Why?



Chapter 1: Detection Theory

What is “Detection”?

Trying to “Detect” things, obviously.

Super.

Chapter 1: Detection Theory

What is “Detection”?

Detection is piecing together *indicators* to determine *something* has happened.

Great..

What the hell is an *indicator*?

Chapter 1: ~~Detection Theory~~ Indicators

A single event, by itself, may not represent anything

Example: Successful login for Bob Loblaw

```
{
  "actor": {
    "alternateId": "bloblaw@bloblaw.blog",
    "displayName": "Okta System",
    "id": "934vcnssifsdhf49jsfd",
    "type": "SystemPrincipal"
  },
  "authenticationcontext": {
    "authenticationStep": 0,
    "externalSessionId": "asjfh948Faijf9jf4Af"
  },
  "client": {},
  "debugcontext": {
    "debugData": {
      "attributesAdded": "",
      "attributesDeleted": "",
      "attributesModified": "",
      "dtHash": "jwoca9rvusajfh4hfjvnf4ovtjnes8tvnc1444whvnaekthvenskghsvntiueh"
    }
  },
  "displaymessage": "Successful login",
  "eventtype": "user.session.start",
  "legacyeventtype": "user.session.start",
  "outcome": {
    "result": "SUCCESS"
  },
  "published": "2022-08-04 19:36:42.729",
  "request": {},
  "securitycontext": {
    "asNumber": 812,
    "asOrg": "bloblaw law",
    "domain": "bloblaw.blog",
    "isProxy": false,
    "isp": "bloblaw law."
  },
  "severity": "INFO",
  "target": [
    {
      "alternateId": "unknown",
      "detailEntry": {
        "direction": "Browser to Okta",
        "sourceProfile": "fjejf9w4jfaifjjaeifjf",
        "targetProfile": "f489jfaifjaofnadskj"
      },
      "displayName": "fjew4aifja0jfaeiuigaskjdhfaskhf",
      "id": "pwofjeoiarjgfodin",
      "type": "ProfileMapping"
    }
  ]
},
  "transaction": {
    "detail": {},
    "id": "unknown"
  },
  "uuid": "fcjcaofj-144a-13cd-b346-sadfsdfb4548ee1f",
  "version": "0"
}
```

Chapter 1: ~~Detection Theory~~ Indicators

Multiple single events, over time, may be far more nefarious

Example: 4 failed login attempts over 15 minutes then a successful login

+1 low/no rep IP

+1 unknown device

+1 failed login

+1 failed login

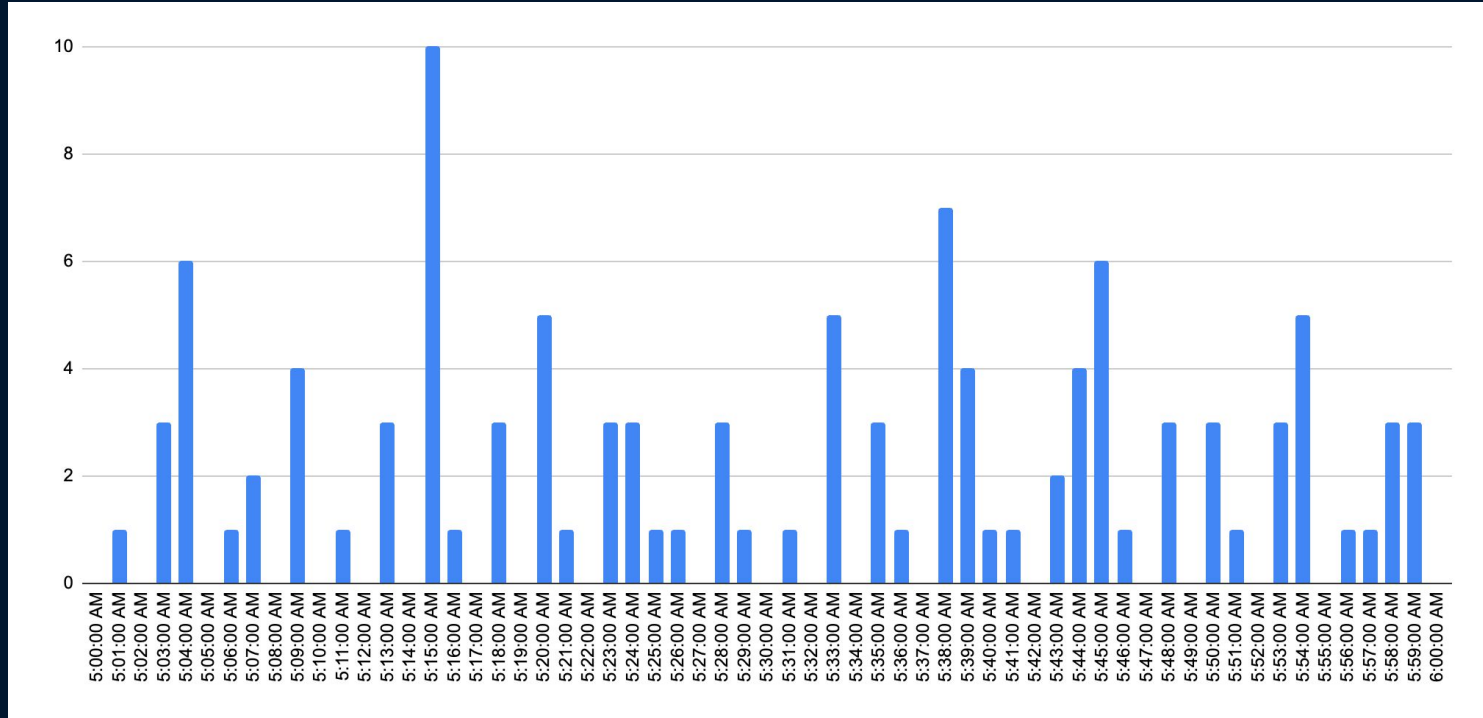
+1 failed login

+1 failed login

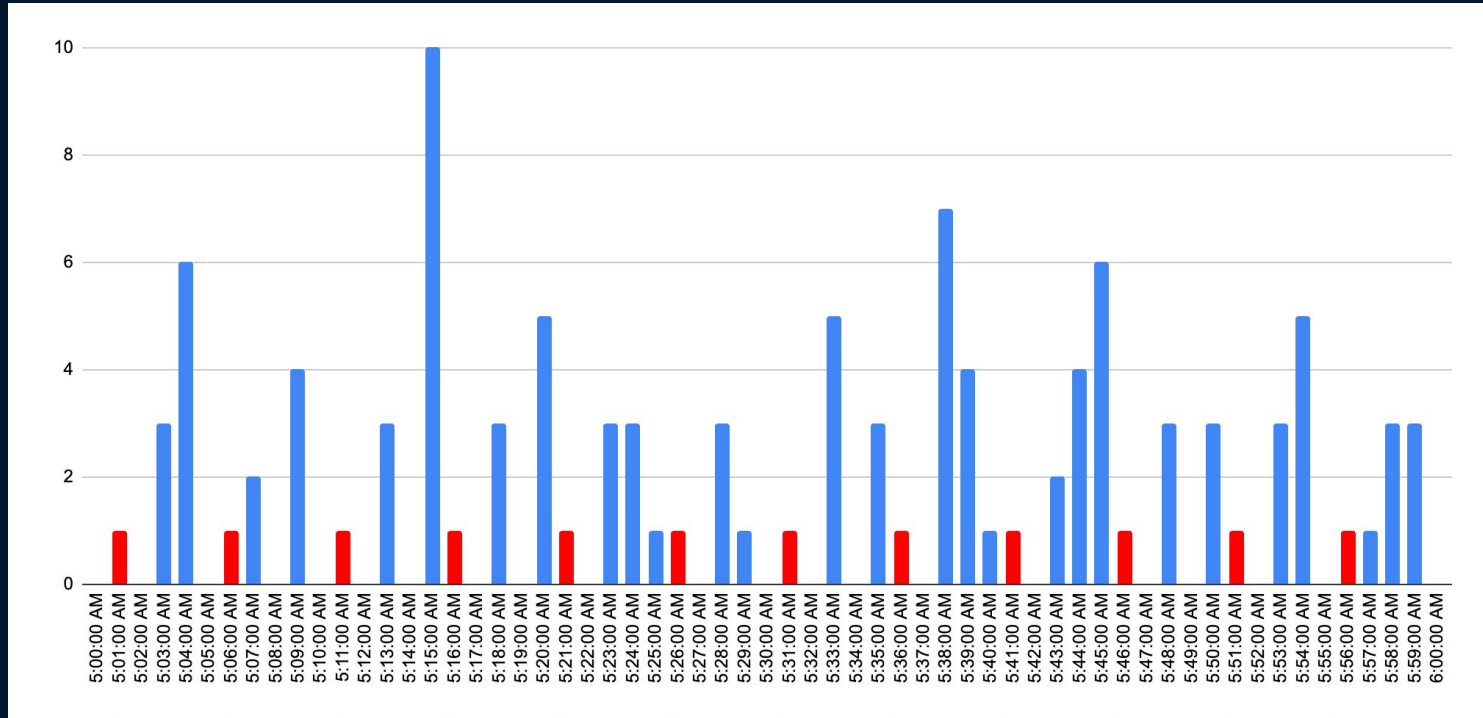
successful login

Suspicious?

What about beaconing?



What about beaconing?



Chapter 1: Detection Theory

What is “Detection”?

Trying to spot a needle or three(ish) in a stack of needles that:

- Vary in value for various subjective reasons
- The quality depend on dev's (largely) not sure how or why they write to log anyways
- May be produced by well funded bad actors actively attempting to avoid detection.

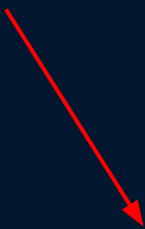
Chapter 2: Meaningful Alerting

First, what's the point of alerting?

Chapter 2: Meaningful Alerting

First, what's the point of alerting?

1. Tell you an *event* was detected



```
Unauthorized Production Secret Accessed
```

```
User: Bob Loblaw
```

```
Secret: Super Secret Secret Key
```

```
AWS Account: bloblaw-logblog-prod (34894817398)
```

```
Timestamp: 20241018 18:42 UTC
```

```
JIRA: SIRT-81234
```

Chapter 2: Meaningful Alerting

First, what's the point of alerting?

1. Tell you an *event* was detected
2. A summary of the finding

```
Unauthorized Production Secret Accessed  
User: Bob Loblaw  
Secret: Super Secret Secret Key  
AWS Account: bloblaw-logblog-prod (34894817398)  
Timestamp: 20241018 18:42 UTC  
JIRA: SIRT-81234
```

Chapter 2: Meaningful Alerting

First, what's the point of alerting?

1. Tell you an *event* was detected

2. A summary of the finding

3. Makes record of it

Unauthorized Production Secret Accessed

User: Bob Loblaw

Secret: `Super Secret Secret Key`

AWS Account: bloblaw-logblog-prod (34894817398)

Timestamp: 20241018 18:42 UTC

JIRA: SIRT-81234

Chapter 2: Meaningful Alerting

But what are we really trying to accomplish?

1. Catch *bad* things

Chapter 2: Meaningful Alerting

But what are we really trying to accomplish?

1. Catch *bad* things
2. React quickly

Chapter 2: Meaningful Alerting

But what are we really trying to accomplish?

1. Catch *bad* things
2. React quickly
3. Mitigating problems before they blow up

Chapter 2: Meaningful Alerting

But what are we really trying to accomplish?

1. Catch *bad* things
2. React quickly
3. Mitigating problems before they blow up (even worse)

Chapter 2: Meaningful Alerting

While not...

Getting inundated by false positives and meaningless noise

Chapter 2: Meaningful Alerting

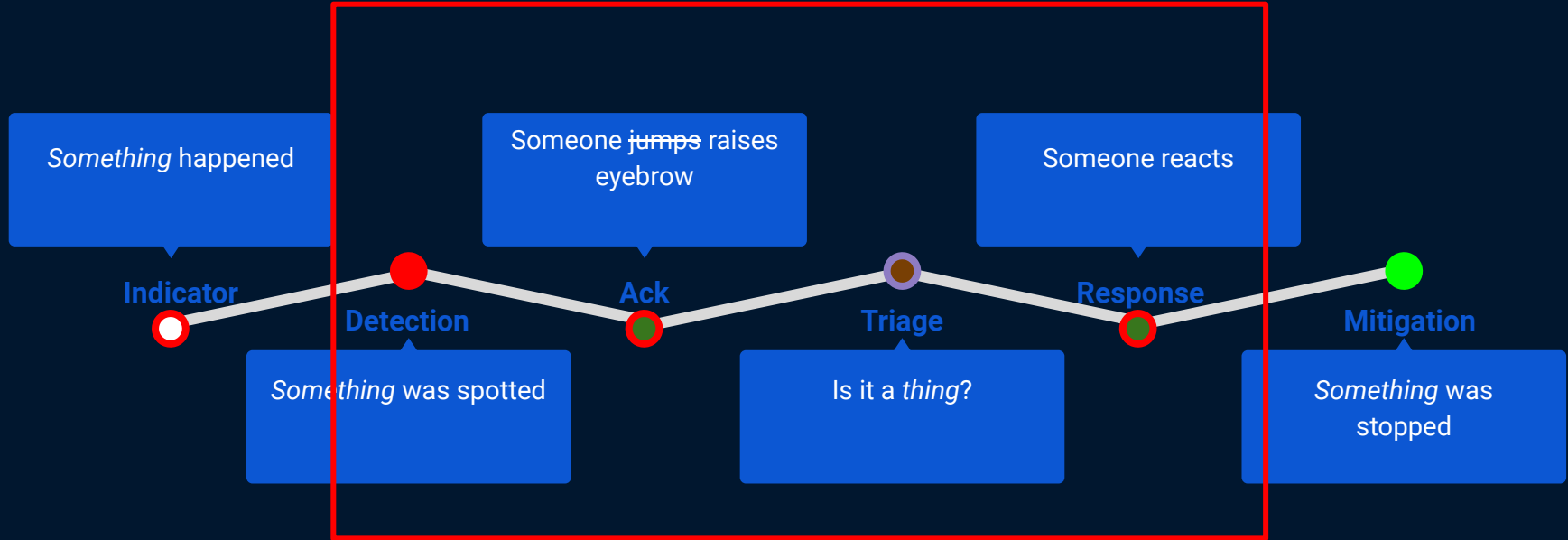
While not...

Getting inundated by false positives and meaningless noise

Or...

Alerts with so little information you have no idea what's going on

What if we can make this better?



Chapter π: Enrichment!

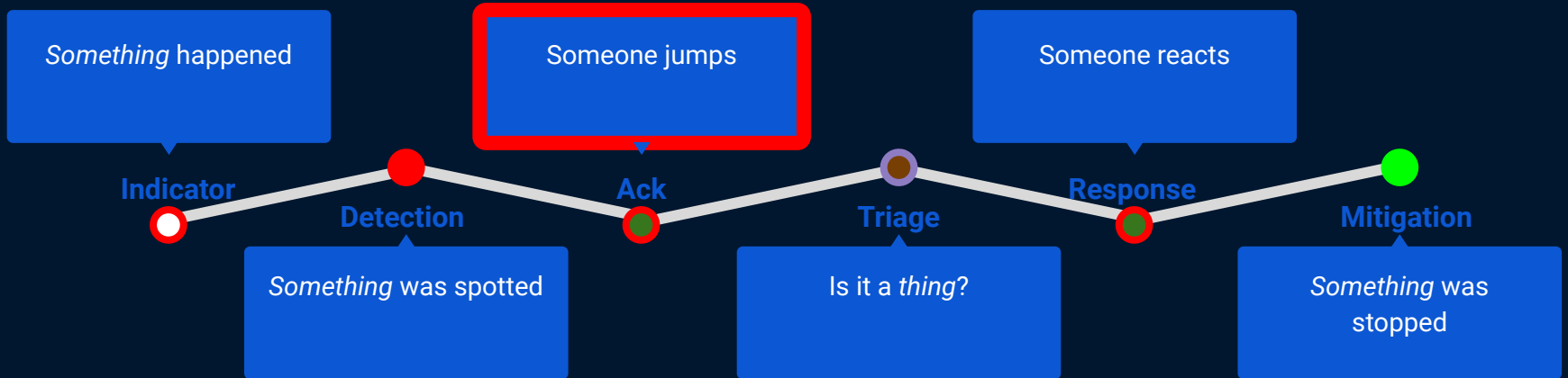
Imagine this:

- AWS, a production account (“bloblaw-blog-prod”)
- SOP changes are done using pull requests; GitHub Actions + Terraform
- Emergency / break-glass access via local IAM user + MFA, user: walbolbob

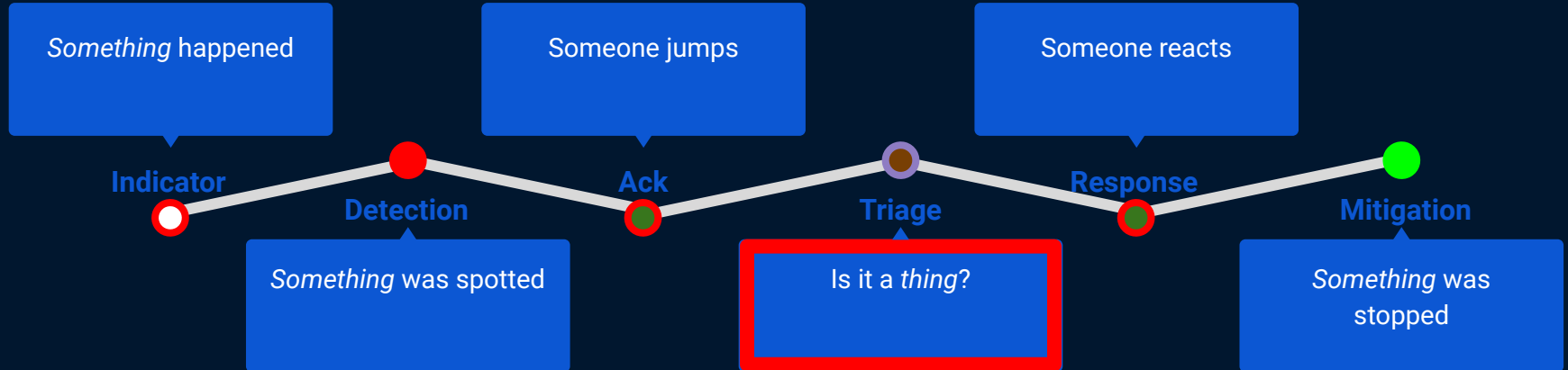
It’s 3:00am, You get paged because walbolbob made a change to an S3 bucket’s policy in bloblaw-blog-prod. The bucket: bloblaw-blog-blob



So here we are...



Barrelling towards...



Triage!

S3 Bucket Policy Modified by Local IAM

User: walbolbob

Bucket: bloblaw-blog-blob

AWS Account: bloblaw-blog-prod (34894817398)

Timestamp: 2024-10-31T08:04:00Z

JIRA: [SIRT-23484](#)

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "IEEEIIU74NPAV3K76",
    "arn": "arn:aws:iam::34894817398:user/walbolbob",
    "accountId": "34894817398",
    "accessKeyId": "AAAAEEEEIIU74NPAV3KK76",
    "userName": "walbolbob",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "true",
        "creationDate": "2024-10-31T08:02:00Z"
      }
    }
  },
  "eventTime": "2024-10-31T08:04:00Z",
  "eventSource": "s3.amazonaws.com",
  "eventName": "PutBucketPolicy",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "172.16.165.101",
  "userAgent": "Mozilla",
  "requestParameters": {
    "host": ["s3.amazonaws.com"],
    "bucketName": "bloblaw-blog-blob",
    "bucketPolicy": {
      "Id": "",
      "Statement": [
        {
          "Action": "s3:*",
          "Effect": "Allow",
          "Principal": {
            "AWS": [
              "arn:aws:iam::34894817398:user/*"
            ],
            "Resource": [
              "arn:aws:s3:::bloblaw-blog-blob",
              "arn:aws:s3:::bloblaw-blog-blob/*"
            ]
          }
        }
      ]
    },
    "Version": "2012-10-17"
  },
  "responseElements": null,
  "additionalEventData": {
    "SignatureVersion": "SigV4",
    "CipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
    "AuthenticationMethod": "AuthHeader"
  },
  "requestID": "1",
  "eventID": "1",
  "eventType": "AwsApiCall",
  "recipientAccountId": "34894817398"
}
```

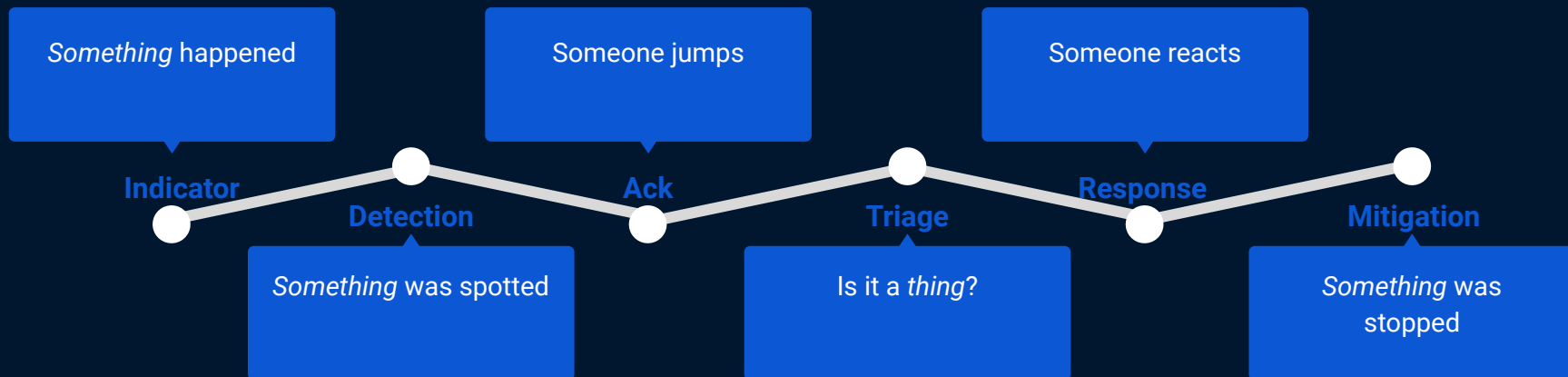
Triage!

- Is this actually Bob's IP?
- What's he doing up at 3am?
- Is his device online?
- Why's he using that emergency account?
- What else has that IP been up to?

```
"eventVersion": "1.05",
"userIdentity":
{
  "type": "IAMUser",
  "principalId": "IEEEIIIU74NPAV3K76",
  "arn": "arn:aws:iam::34894817398:user/walbolbob",
  "accountId": "34894817398",
  "accessKeyId": "AAAAEEEEIIIU74NPAV3KK76",
  "userName": "walbolbob",
  "sessionContext":
  {
    "attributes":
    {
      "mfaAuthenticated": "true",
      "creationDate": "2024-10-31T08:02:00Z"
    }
  }
},
"eventTime": "2024-10-31T08:04:00Z",
"eventSource": "s3.amazonaws.com",
"eventName": "PutBucketPolicy",
"awsRegion": "us-west-2",
"sourceIPAddress": "172.16.165.101",
"userAgent": "Mozilla",
"requestParameters":
{
  "host": ["s3.amazonaws.com"],
  "bucketName": "bloblaw-blog-blob",
  "bucketPolicy": {
    "Id": "",
    "Statement": [
```


Time's ticking!

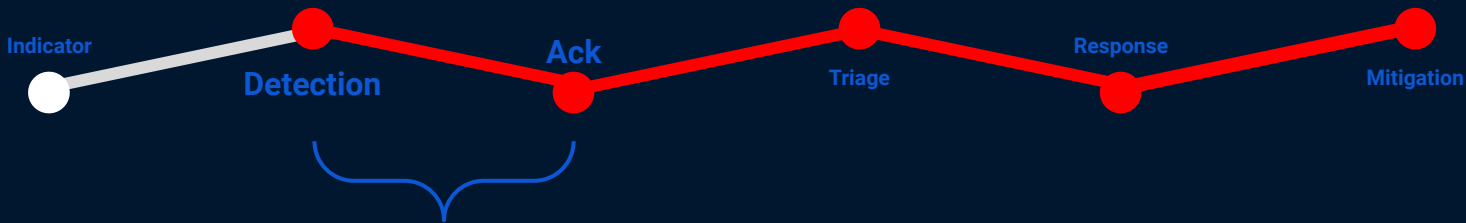
Mean-time-to-Mitigate (MTTM)



Mean-time-to-Mitigate (MTTM)

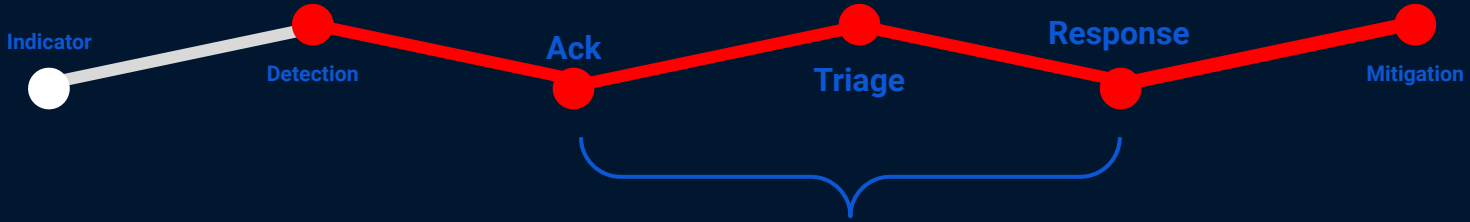


Mean-time-to-Mitigate (MTTM)



- Accuracy of detection rules
- Effectiveness of determining initial severity
- If/when to page on-call

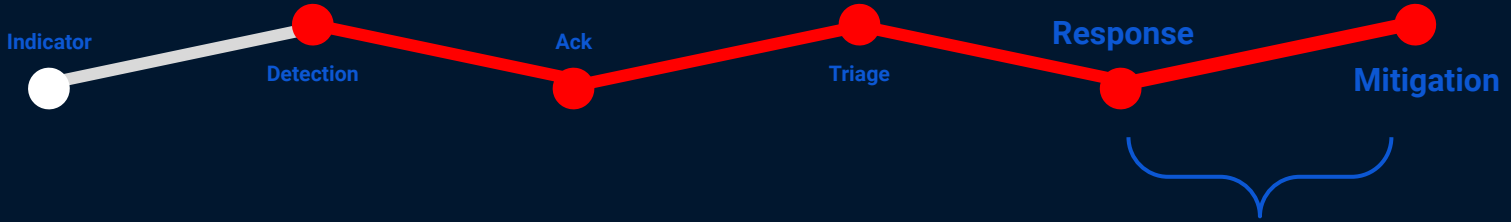
Mean-time-to-Mitigate (MTTM)



- Verbose, meaningful data
- Easy to read and comprehend
- Anticipating the first steps of response

Answer before Asked

Mean-time-to-Mitigate (MTTM)



- Common mitigation best practices
- Understanding what data is required to decide direction
- Easy access to runbooks and tooling

How would this work?

Well...

Our tools: Panther Labs, Tines, Mattermost

- Panther Labs - panther.com
- Tines - tines.com
- Mattermost - mattermost.com (obviously)

Panther Labs

Detection as Code

```
S3_POLICY_CHANGE_EVENTS = {"PutBucketPolicy", "DeleteBucketPolicy"}
PROD_ACCOUNTS = {"34894817398"}
MONITOR_BUCKETS = {"bloblaw-blog-blob"}

def rule(event):
    return (
        event.get("eventName") in S3_POLICY_CHANGE_EVENTS
        and not (event.get("errorCode", "") or event.get("errorMessage", ""))
    )

def title(event):
    return f"S3 bucket modified by [{event.deep_get('userIdentity', 'arn')}]"
```

```
def alert_context(event):
    return {
        "sourceIPAddress": event.get("sourceIPAddress"),
        "bucketName": event.deep_get("requestParameters", "bucketName"),
        "userArn": event.deep_get("userIdentity", "arn"),
        "recipientAccountId": event.get("recipientAccountId"),
        "eventTime": event.get("eventTime")
    }

def severity(event):
    if event.get("recipientAccountId") in PROD_ACCOUNTS:
        if event.deep_get("requestParameters", "bucketName") in MONITOR_BUCKETS:
            return "High"
        return "Medium"
    return "Low"
```

Panther Labs

Detection as Code

Detection rule/instruction

Customizable event output

Baseline severity

```
S3_POLICY_CHANGE_EVENTS = {"PutBucketPolicy", "DeleteBucketPolicy"}
PROD_ACCOUNTS = {"34894817398"}
MONITOR_BUCKETS = {"bloblaw-blog-blob"}

def rule(event):
    return (
        event.get("eventName") in S3_POLICY_CHANGE_EVENTS
        and not (event.get("errorCode", "") or event.get("errorMessage", ""))
    )

def title(event):
    return f"S3 bucket modified by [{event.deep_get('userIdentity', 'arn')}]"

def alert_context(event):
    return {
        "sourceIPAddress": event.get("sourceIPAddress"),
        "bucketName": event.deep_get("requestParameters", "bucketName"),
        "userArn": event.deep_get("userIdentity", "arn"),
        "recipientAccountId": event.get("recipientAccountId"),
        "eventTime": event.get("eventTime")
    }

def severity(event):
    if event.get("recipientAccountId") in PROD_ACCOUNTS:
        if event.deep_get("requestParameters", "bucketName") in MONITOR_BUCKETS:
            return "High"
        return "Medium"
    return "Low"
```

Triage!

- Is this actually Bob's IP?
- What's he doing up at 3am?
- Is his device online?
- Why's he using that emergency account?
- What else has that IP been up to?

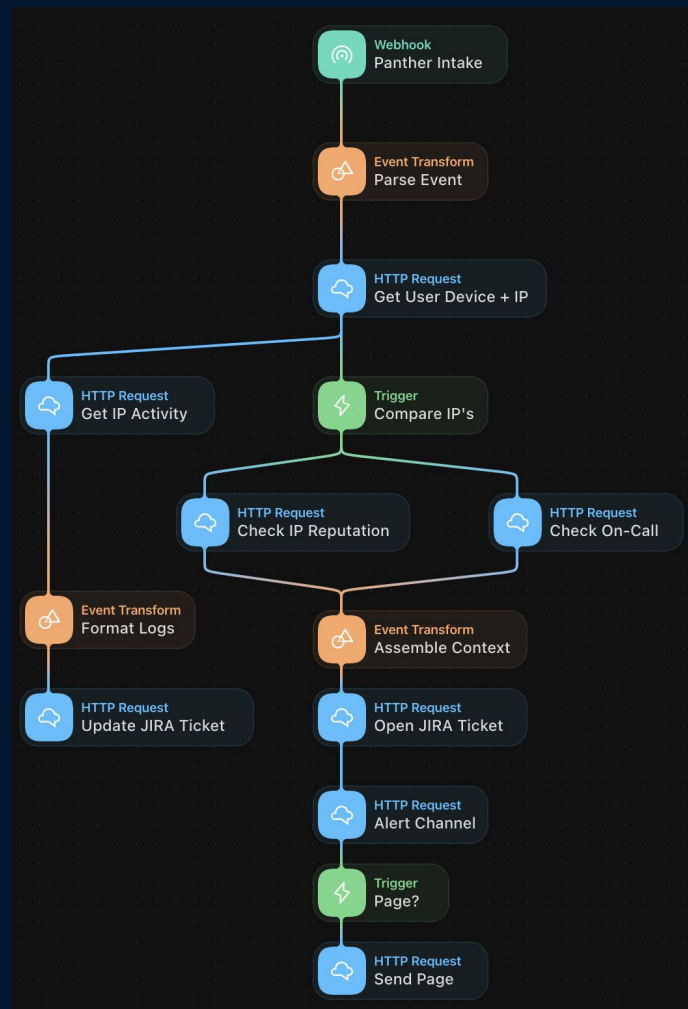
```
"eventVersion": "1.05",
"userIdentity":
{
  "type": "IAMUser",
  "principalId": "IEEEIIIU74NPAV3K76",
  "arn": "arn:aws:iam::34894817398:user/walbolbob",
  "accountId": "34894817398",
  "accessKeyId": "AAAAEEEEIIIU74NPAV3KK76",
  "userName": "walbolbob",
  "sessionContext":
  {
    "attributes":
    {
      "mfaAuthenticated": "true",
      "creationDate": "2024-10-31T08:02:00Z"
    }
  }
},
"eventTime": "2024-10-31T08:04:00Z",
"eventSource": "s3.amazonaws.com",
"eventName": "PutBucketPolicy",
"awsRegion": "us-west-2",
"sourceIPAddress": "172.16.165.101",
"userAgent": "Mozilla",
"requestParameters":
{
  "host": ["s3.amazonaws.com"],
  "bucketName": "bloblaw-blog-blob",
  "bucketPolicy": {
    "Id": "",
    "Statement": [
```

Tines

“Story” driven

Drag-and-drop automation

Infinitely customizable



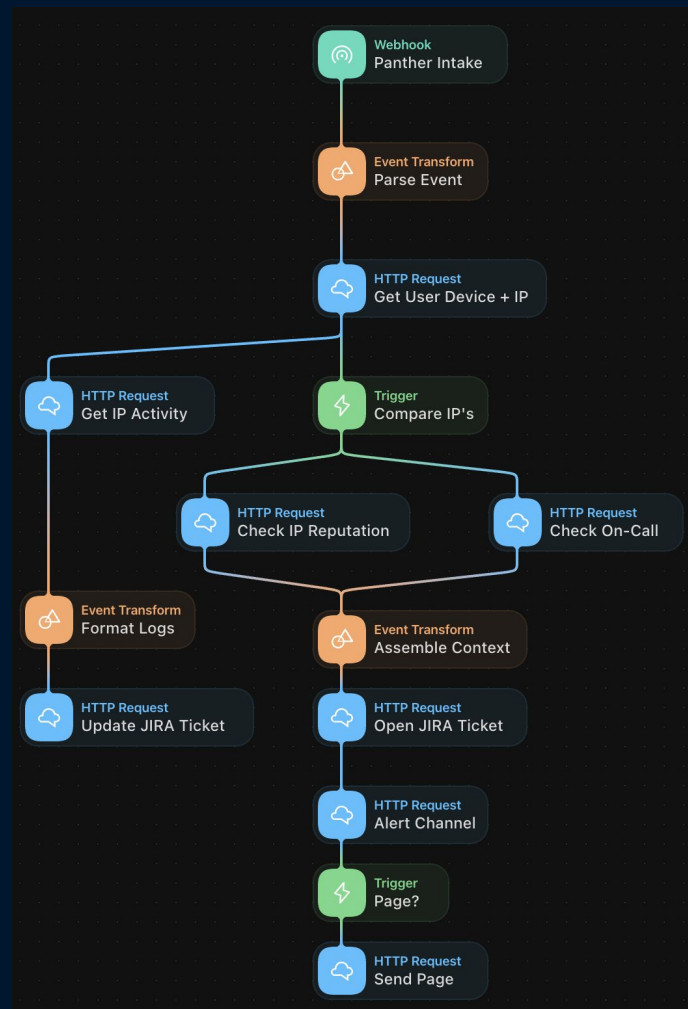
Tines

“Story” driven

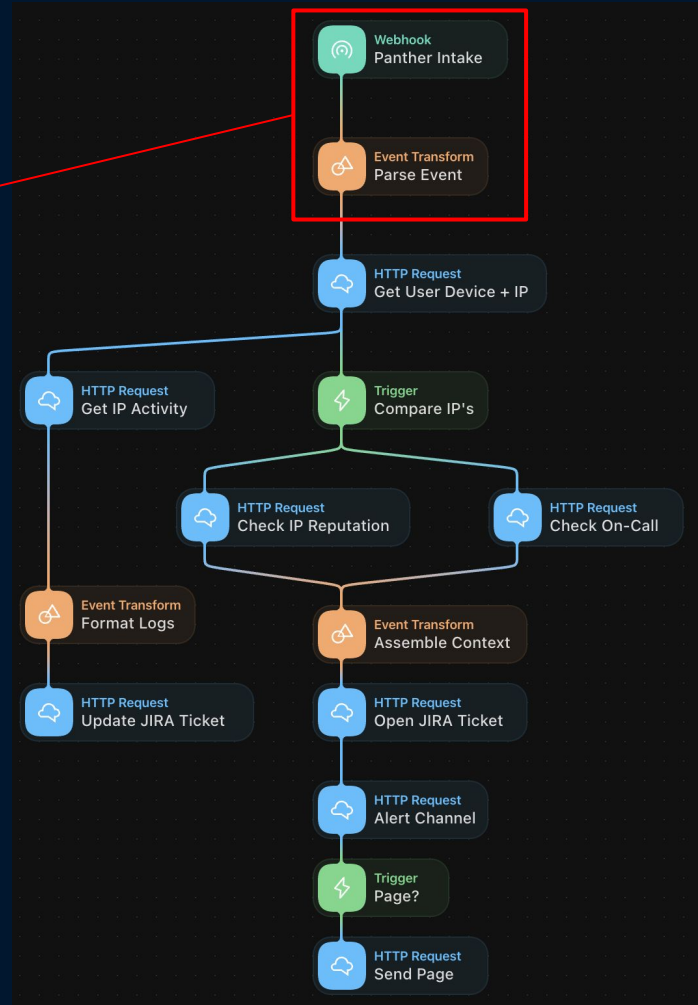
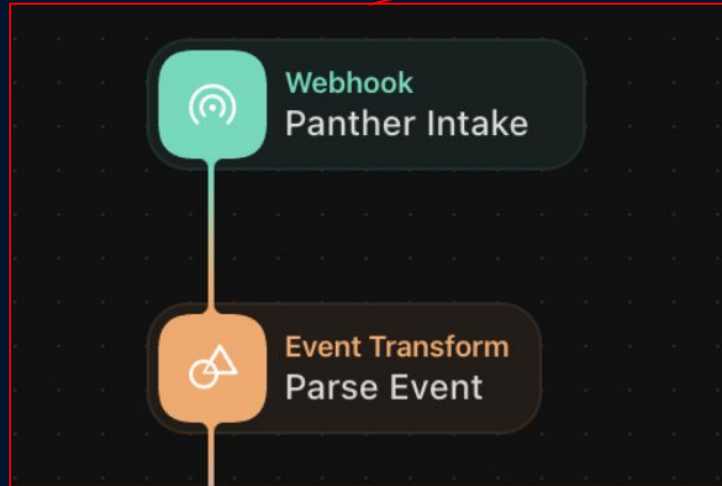
Drag-and-drop automation

Infinitely customizable

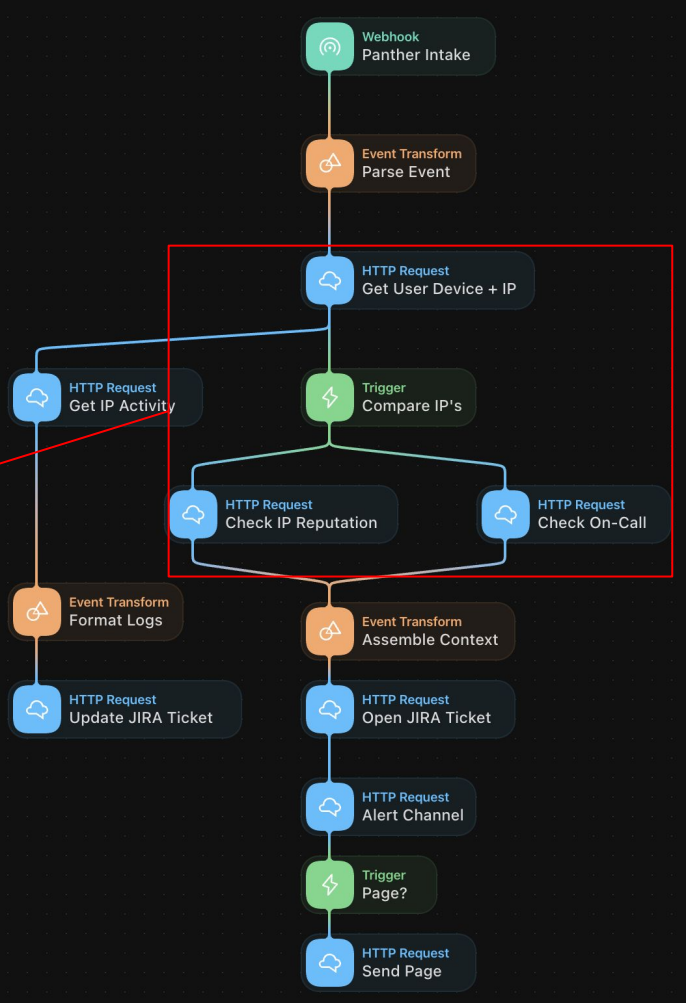
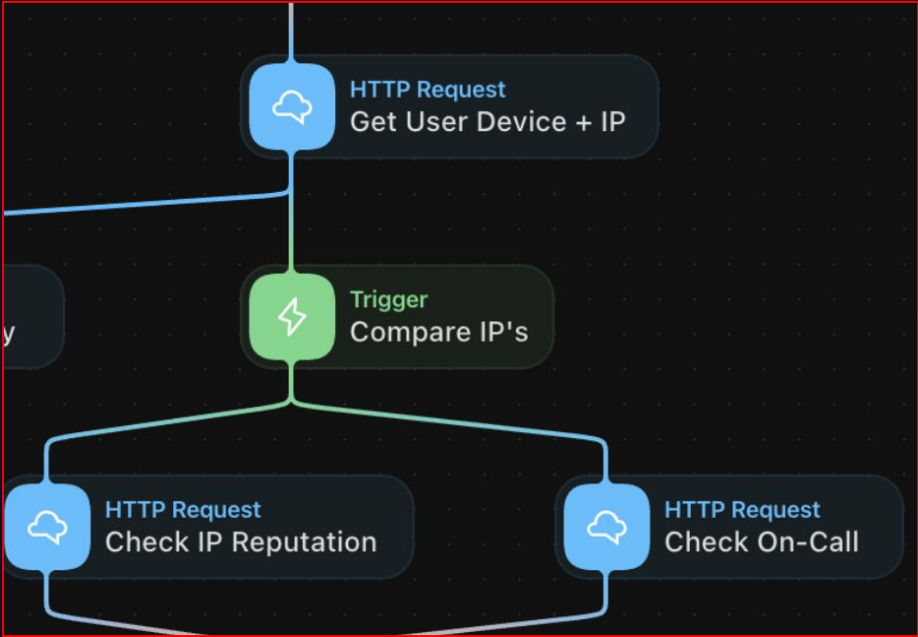
Very addictive



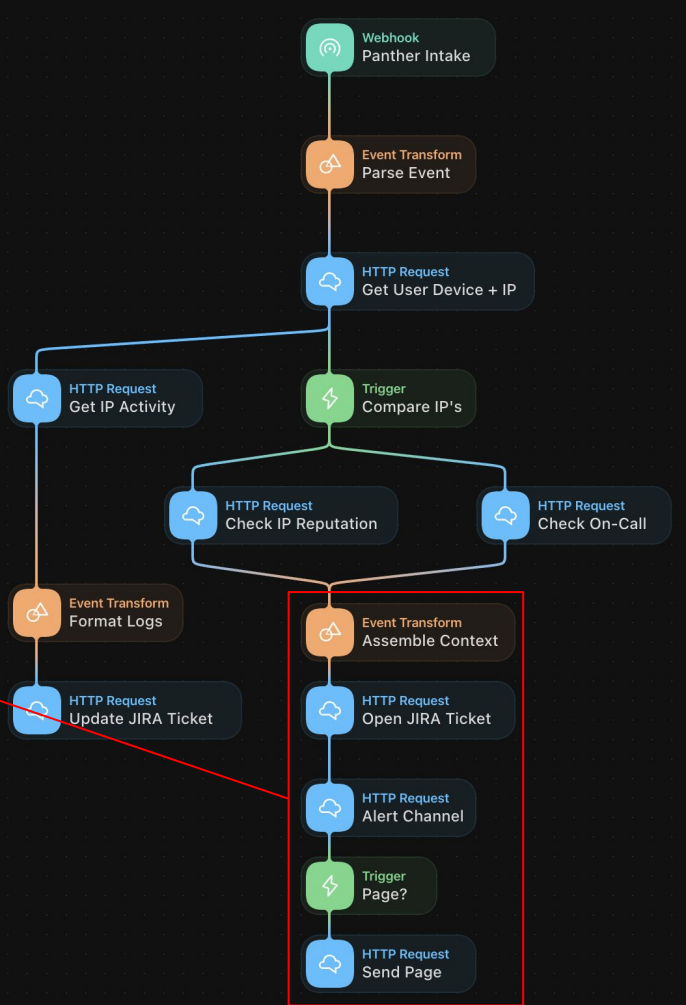
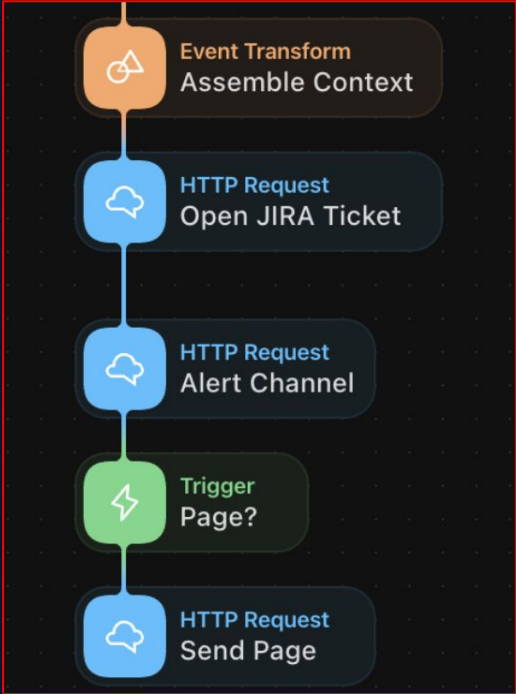
Intake



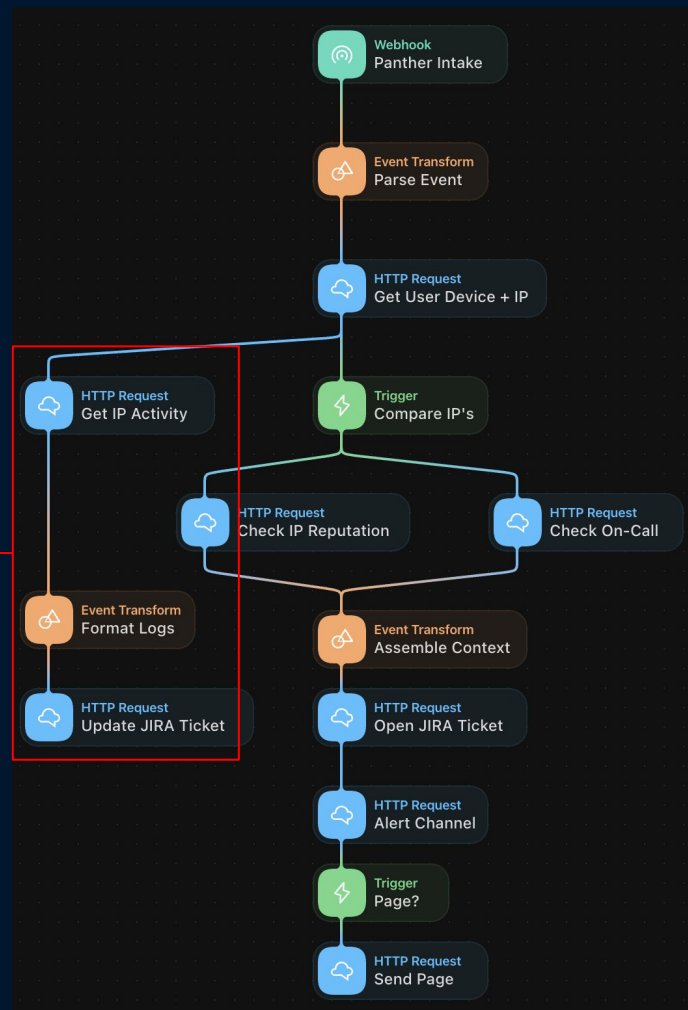
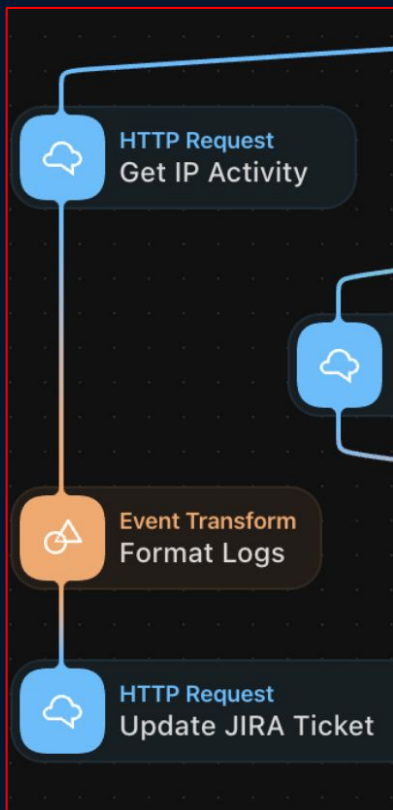
Enrichment

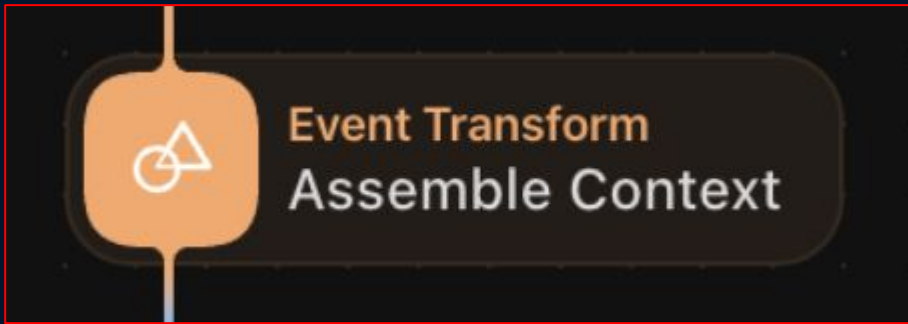


Delivery



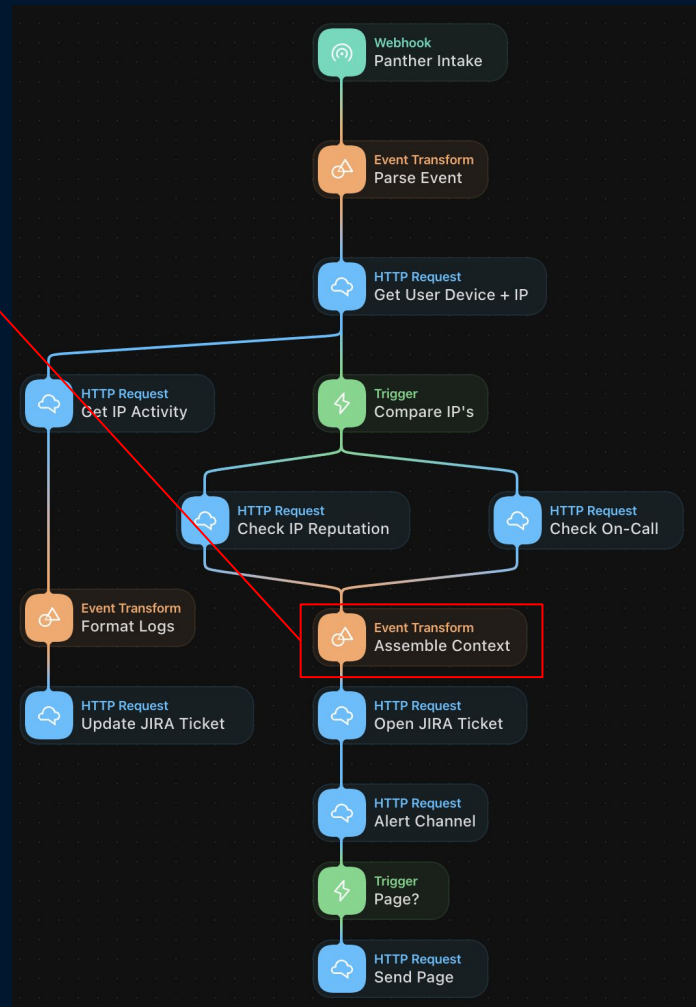
Gravy





Where decisions are made...

- What severity should this be?
- Should we page?
- Should we even alert?
- Bolt-on scoring mechanisms
- Exception rules
- IOC lists



Tines

Alert destination webhook

Getting User IP and device data

Collect IP activity logs*

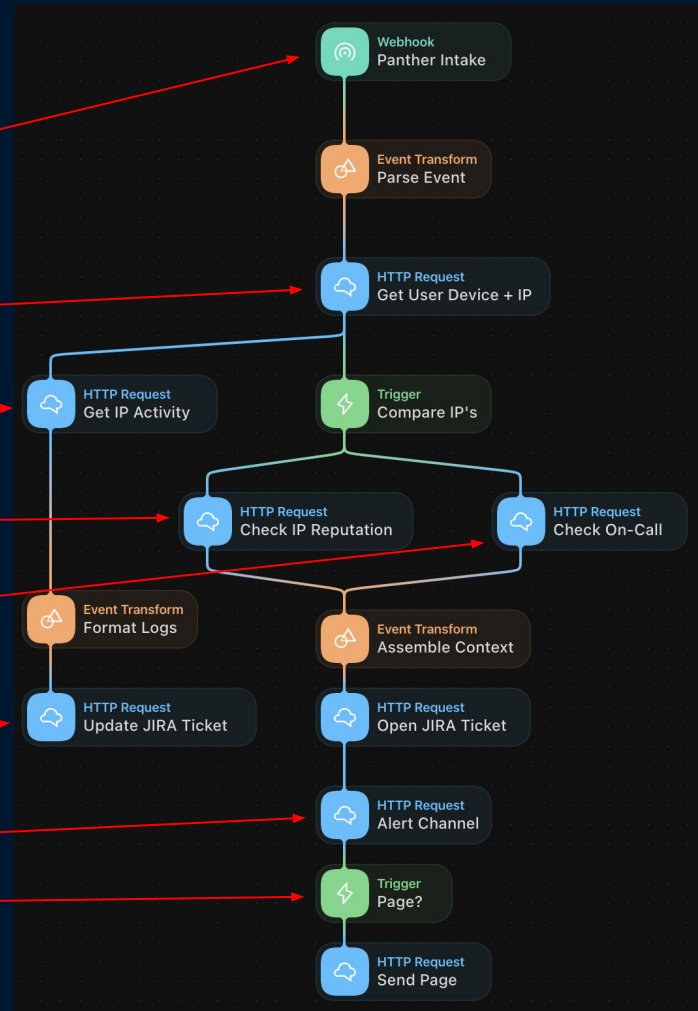
Check IP Reputation

Checking if they're on-call + active events

Tickets

Notifies

Pages as necessary!



Mattermost

S3 bucket modified by [`arn:aws:iam:34894817398:user/walbolbob`]

User: walbolbob

Bucket: bloblaw-blog-blob

AWS Account: `bloblaw-blog-prod` (34894817398)

Timestamp: 2024-10-31T08:04:00Z

IP Check: Passed

Device Check: Passed

User On-Call: True

Severity: Medium

JIRA: [SIRT-23484](#)

Where are we now?



```
S3 bucket modified by [ arn:aws:iam:34894817398:user/walbolbob ]
User: walbolbob
Bucket: bloblaw-blog-blob
AWS Account: bloblaw-blog-prod (34894817398)
Timestamp: 2024-10-31T08:04:00Z
IP Check: Passed
Device Check: Passed
User On-Call: True
Severity: Medium
JIRA: SIRT-23484
```

- 1 - High confidence of User
 - “Regular” IP
 - Device in Use
- 2 - User is On-Call
 - 3am activity
 - Hand-bombed change
- 3 - Access to logs in JIRA ticket
 - Pre-prepared for (near) immediate consumption

Security Incident => Policy Incident

Questions?



info@DevSecOops.ca