



**APRIL 3-4, 2025**  
BRIEFINGS

# Determining Exploitability of Vulnerabilities with SBOM and VEX

Anusha Penumacha  
Srinija Kammari



Software Composition  
Analysis



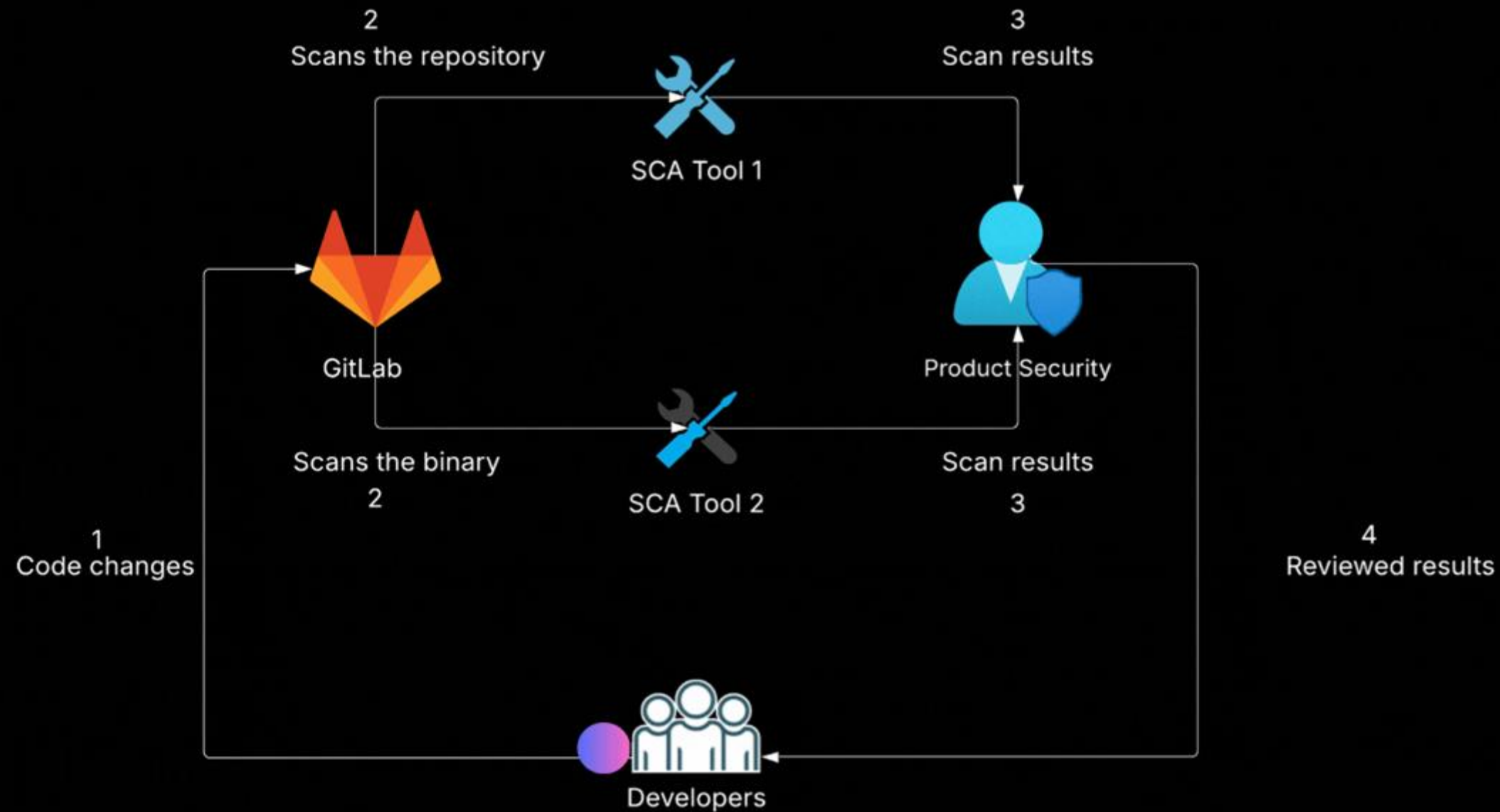
Software Bill Of Materials



Vulnerability Exploitability  
eXchange



# Our Open-Source Security Posture ~ 2022



## Our Open-Source Security Posture ~ 2022

Self Service, developer  
friendly

Shift left mechanism  
enabled developer  
ownership

Difficult to gather org-  
wide inventory

No holistic picture for  
Product Security

# Here are some things we did before building telemetry

## Building Asset Inventory

Source of Truth for  
the product  
(repositories/artifacts)

## Building Centralized scan platform

Scalable centralized  
system for performing  
end-to-end operations

# Step 1: Build an asset inventory + ownership mapping

- ☐ Executive Order for SBOM came in as an opportunity
- ☐ Emphasized the importance of asset inventory management
- ☐ Started mapping repos/artifacts towards products
- ☐ Need to build internal tooling to maintain this but necessary



# CTS Portal

Welcome, Srinija Kammari!

Choose one of the options below to continue:

## Repository Onboarding & Offboarding

Onboarding

Gitlab Group  
Onboarding

Offboarding

## Artifact Onboarding & Offboarding

Onboarding

Offboarding

## More CTS Info

CTS Docs

Onboarding Help

CTS Features

# CTS Portal

## CTS Repository Onboarding

ONBOARDING FORM:

The full url for your repository. Answer must start with "https://" and should not end with a slash. \*

\* = required

E.g.

Which branch are you onboarding? (For Github repos, you can only choose the first option) \*

- ☒ Specific branch(es). For multiple branches, please separate them with a comma.  
E.g. main, develop, release123

- ☐ Default branch (Recommended option for group onboarding)

Who would you like the tickets to be assigned to? \*

- ☒ Specific assignees. Please specify exactly 3 users in order, separated by a comma.  
E.g. user1@splunk.com, user2@splunk.com, user3@splunk.com

- ☐ Most recent committer to the branch  
☐ The developer who first made the commit (to introduce the vulnerability)  
☐ Privileged user from the CODEOWNERS file

The jira watcher(s) you would like set on your tickets. For multiple users, please separate them with a comma (max 3).

The JIRA label(s) that you want on your tickets. For multiple labels, separate them with a comma (max 3).

The product area that you would like set on your tickets: \*

The mission team that you would like set on your tickets:

The ████ program team that you would like set on your tickets:

The release version that you would like set on your tickets:

Would you like to opt-in to **full ticketing** (ticketing for low/medium severities)? \*

⚠ [NOTE]: Except for Product Area and "Full Ticketing opt-in", all dropdown values are optional. If any dropdowns are missing the values you're looking for, please STOP and reach out on #cvm-sbom-queries to get those values added before continuing. It is important that these values are accurate for your tickets to be routed correctly.

< Back

Clear

Continue

# Step 1: Build an asset inventory + ownership mapping

## ☐ Advantages:

- Scalability
- Product-level Telemetry

## ☐ Disadvantages:

- Operational costs
- Keeping it up-to-date



## Step 2: Build a Centralized System

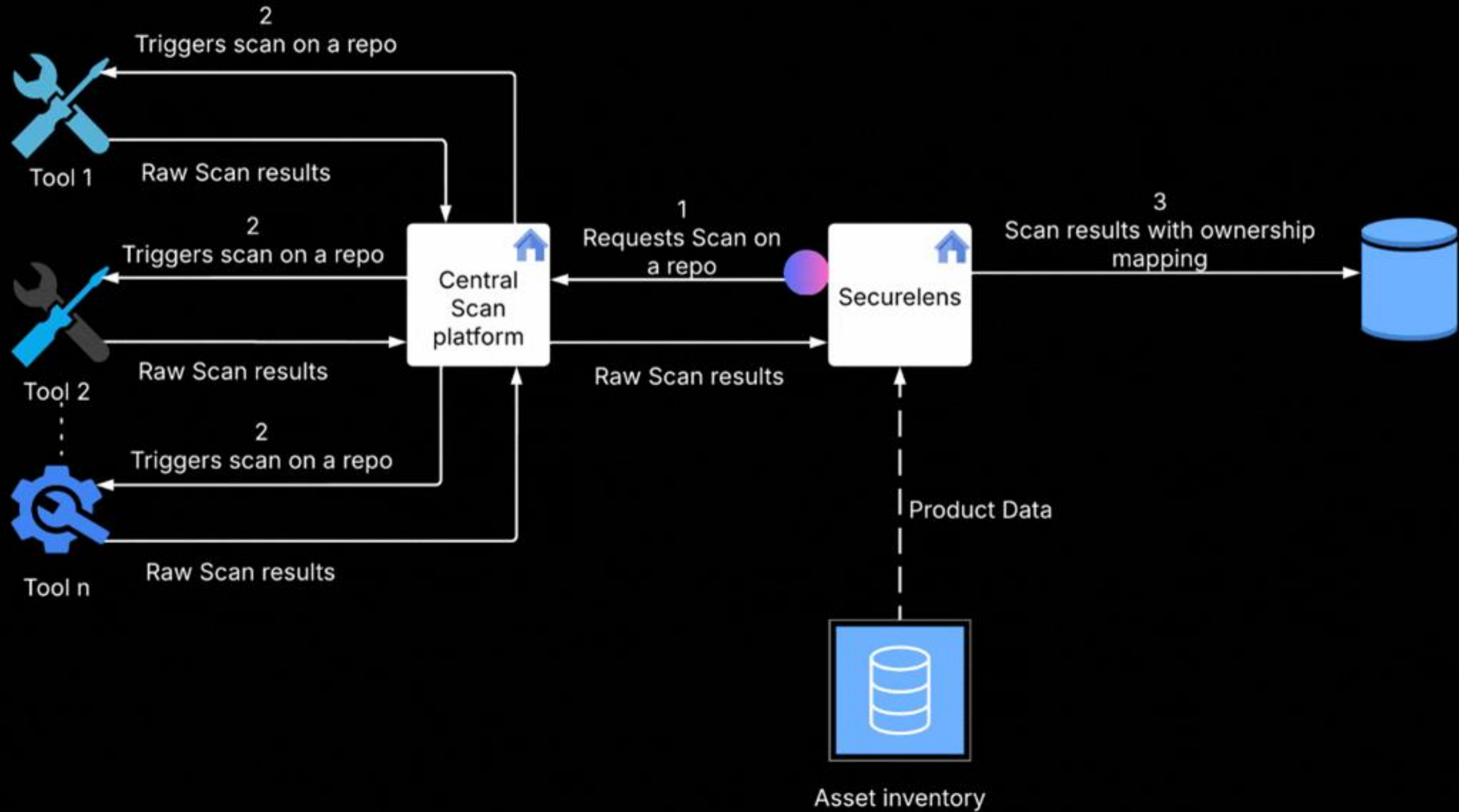
- ☐ Easy to scan, less involvement of Product teams
- ☐ Processes scan results
- ☐ Creates tickets, assigns and notifies owners





**2000 YEARS  
LATER**







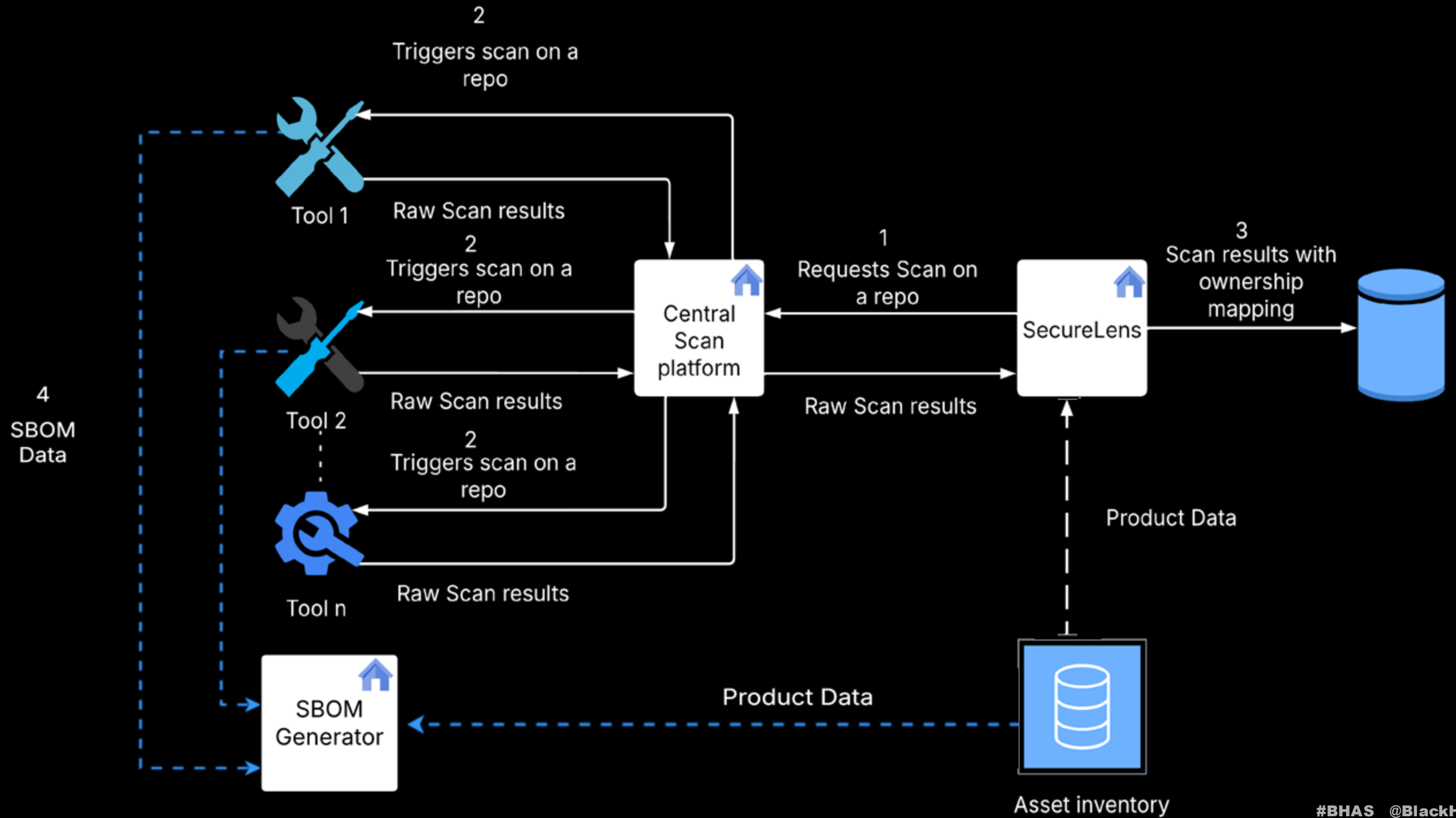
## Step 2: Build a Centralized System

### ☐ Advantages:

- Plug-n-play mechanism
- Not purely relying on CI/CD for scan results

### ☐ Disadvantages:

- Less customization
- Delayed feedback to developers



# Security Central

Show Preferences

Product Area

All Product Areas

Show unassigned tickets only

Hide Filters

Give Feedback here!

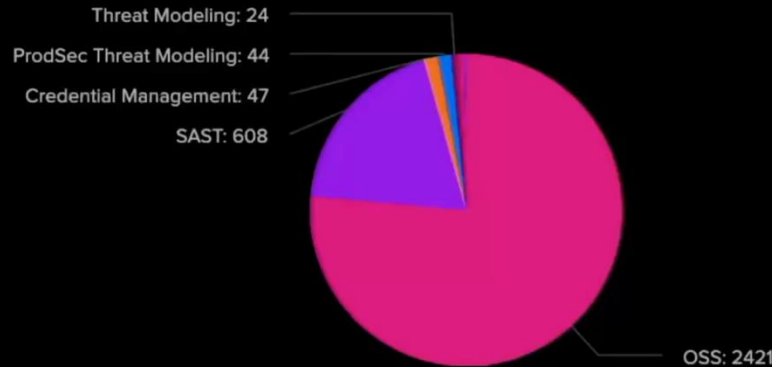
Security Central FAQ

## Product Security

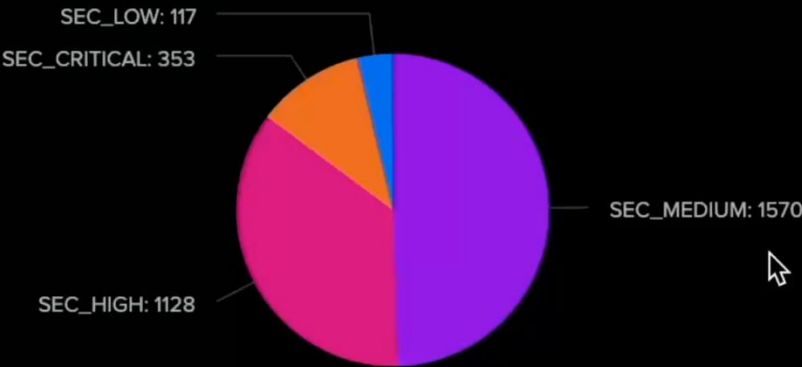
Last Dataset Refreshed at 2025-03-25T09:02:00.000-07:00

Show Findings by Version

Distribution of Findings by Type



Distribution of Findings by Severity



9m ago

Attention Needed

PAST ASSIGNED DUE DATE

SEC_CRITICAL	SEC_HIGH	SEC_MEDIUM	SEC_LOW
224	524	475	62

PAST SLO

SEC_CRITICAL	SEC_HIGH	SEC_MEDIUM	SEC_LOW
232	567	298	9

NEARING SLO

SEC_CRITICAL	SEC_HIGH	SEC_MEDIUM	SEC_LOW
11	459	181	0

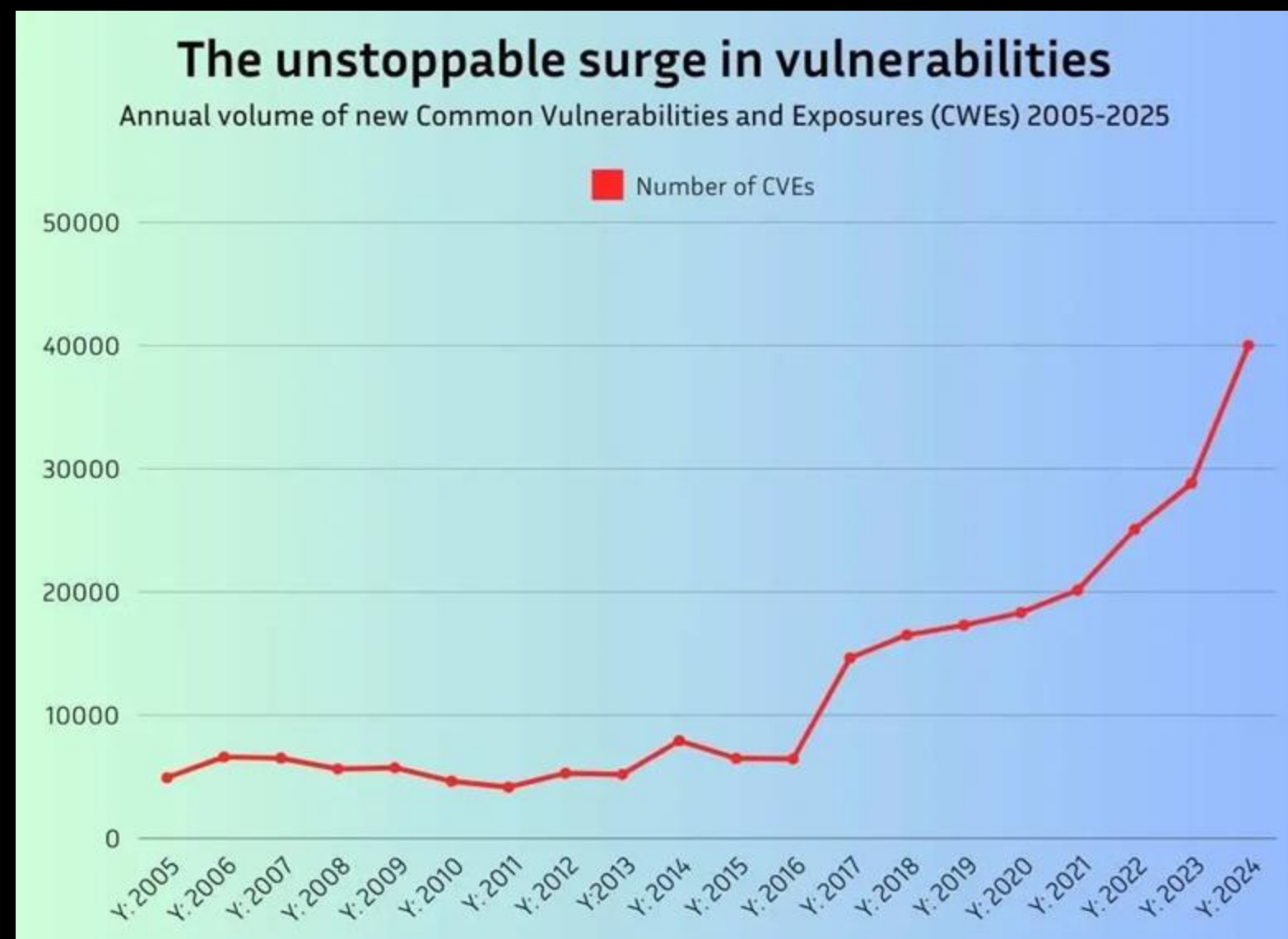


# Disadvantage of ticketing every vulnerability

- ☐ Tools generate a flood of vulnerabilities
- ☐ Not all of them are valid and actionable

Internally, we've discovered that >26% of vulnerabilities found by these tools are non-exploitable / not-fixable.

The CVE count keeps increasing every year.



**How do we handle this flood of vulnerabilities  
and enhance developer experience?**



# Exploitability

Potential or likelihood of a vulnerability being exploited

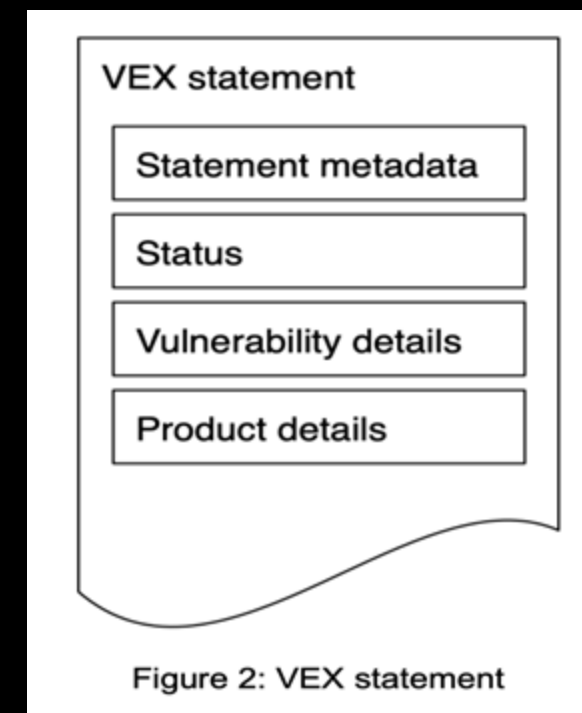
Can be determined using three ways

- ❑ Use publicly available data sources
- ❑ Perform reachability analysis
- ❑ **Ask the developers themselves!!**



# Vulnerability Exploitability Exchange (VEX)

- ❑ A VEX statement indicates the state of a software product or component w.r.t. to the vulnerability. A common VEX use case is to indicate if a software is or is not affected by a vulnerability.
- ❑ Statuses:
  - Not affected
  - Affected
  - Fixed
  - Under Investigation





# Tracking Remediation with VEX

- ❑ Start tracking remediation status of security issues by product
- ❑ Mandate VEX as part of remediation and resolution

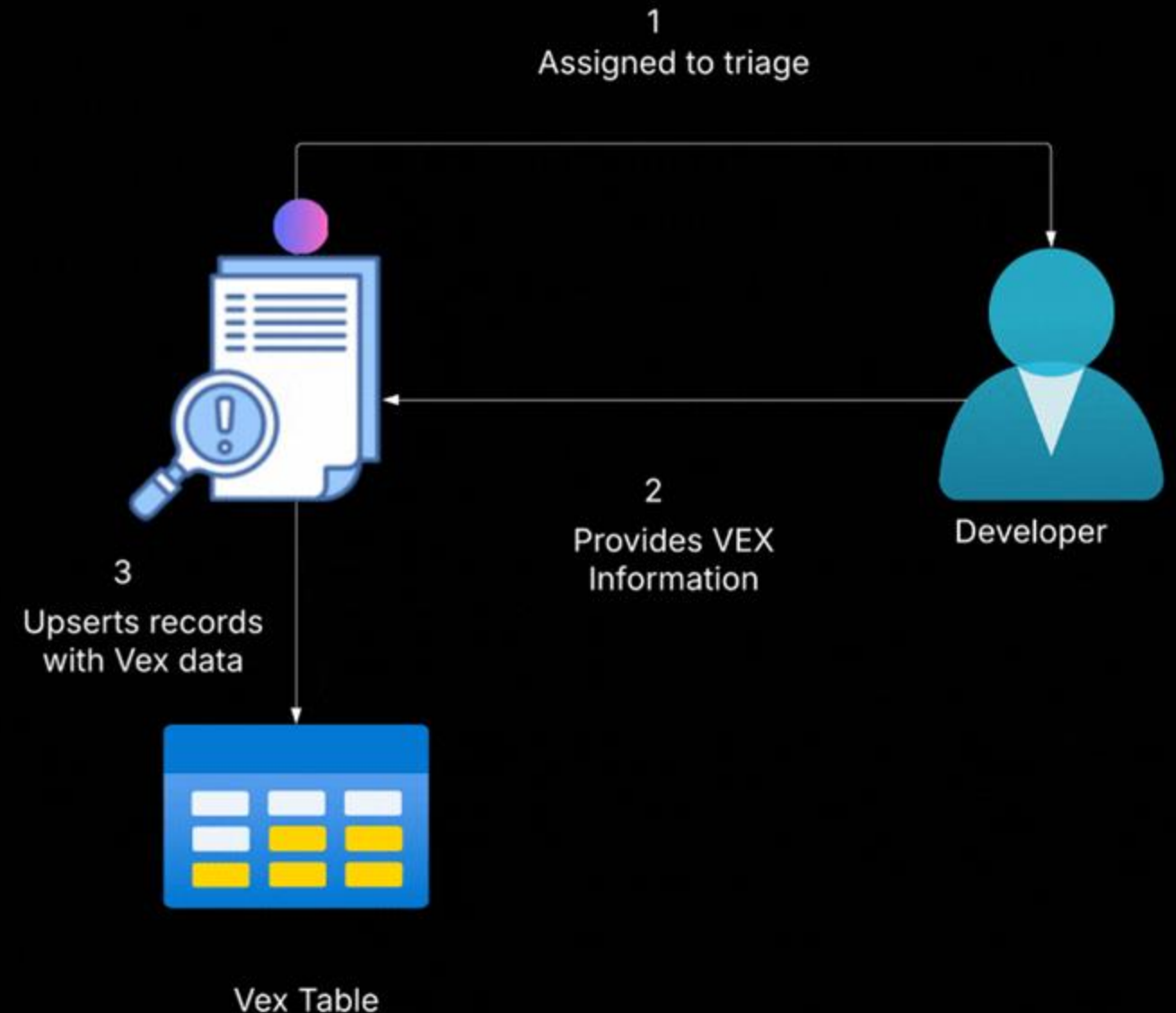




# Developer feedback loop is essential

❑ Cannot purely rely on tool feedback and public sources

- False positives and negatives
- With complex build systems, it's tricky to determine dependencies shipped in the final product.



**Design Choice :**

Why use issue tracking systems to collect this information?

- ❑ Abstraction layer on top of our security scanning and tooling
- ❑ Mandate necessary information that the security teams need. It also works as a medium of back and forth communication
- ❑ Minimal learning curve for Engineering teams
- ❑ Once a generic issue template is agreed upon, it's easy to scale and incorporate additional tooling



# Advantage 1: Getting a clear answer on exploitability

- ❑ Clear answer on exploitability based on usage
- ❑ Allows developer ownership
- ❑ Provides clear audit trail

VEX

Not\_affected

Mandatory fields to be filled: "Vex Status" and

Source

ProdSec

✓ None  
code\_not\_reachable (Not affected directly)  
code\_not\_present (found but not in final product)

VEX

Affected

Mandatory fields to be filled: "Vex Status" and

Source

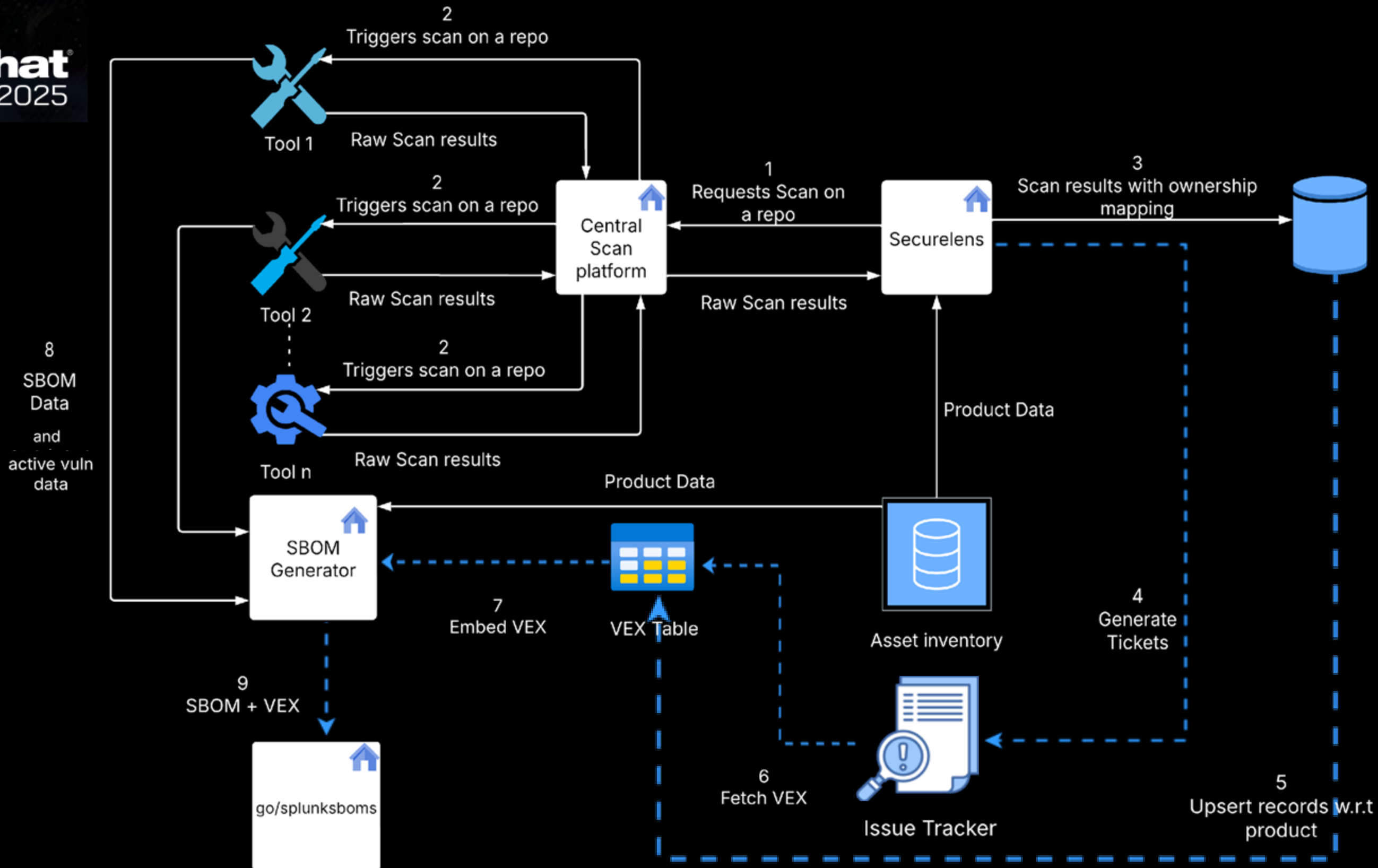
ProdSec

✓ None  
can\_not\_fix (no fix)  
will\_not\_fix(exception)  
fixed\_in\_new\_release(update)  
workaround\_available (custom configuration)

Captures the reporting source for this issue

## Advantage 2 : Automated VEX Generation

```
{
  "affects": [
    {
      "ref": "CVE-2020-23452"
    }
  ],
  "analysis": {
    "firstIssued": "2024-06-12T08:39:55.447-0700",
    "lastUpdated": "2024-06-25T12:11:36.006-0700",
    "state": "resolved"
  },
  "id": "CVE-2020-23452",
  "tools": {
    "components": [
      {
        "name": "Application",
        "type": "application"
      }
    ]
  }
},
{
  "affects": [
    {
      "ref": "CVE-2020-23452"
    }
  ],
  "analysis": {
    "state": "in_triage"
  },
  "id": "CVE-2020-23452",
  "tools": {
    "components": [
      {
        "name": "Application",
        "type": "application"
      }
    ]
  }
},
]
```





## Advantage 3 : Context based Auto Closing and Smart Insights

We try to auto close issues based on previous context (tool agnostic) and also provide some additional insights into the issues that are open.

The key idea is to improve developer experience and reduce triage time

- ❑ **External sources** : We can auto close these issues if we are able to determine from public sources that the vulnerability is not exploitable/cannot be fixed.
- ❑ **Internal sources** :
  - Multiple tools can repeatedly detect an issue that is non exploitable. Auto close any possible future occurrences.
  - Provide additional help on issues based on dependency information and historical data.

Closing ticket because we found related duplicates for the same. Here are the related issues :

○ VULN-23722: XRAY : Upgrade pypi://lxml from [REDACTED] **CLOSED**

○ VULN-23689: XRAY : Upgrade pypi://lxml from [REDACTED] **CLOSED**

○ VULN-23627: XRAY : Upgrade pypi://lxml from [REDACTED] **CLOSED**

○ VULN-19037: XRAY : Upgrade pypi://lxml from [REDACTED]

**CLOSED**

○ VULN-15284: XRAY : Upgrade pypi://lxml from [REDACTED] **CLOSED**

○ VULN-13002: XRAY : Upgrade pypi://lxml from [REDACTED] **CLOSED**



February 3, 2025 at 6:28 PM

Closing ticket because the there is no fix available for the package



S

srv-jira-oss February 3, 2025 at 6:44 PM 

Found some open tickets that have the same path. Please use them for analysis Here are the related issues :

- ☐ [VULN-27166: Container Scan: Upgrade com.google.protobuf:protobuf-java from \[REDACTED\]](#)
  - ☐ [VULN-27155: Container Scan: Upgrade org.apache.avro:avro from \[REDACTED\]](#)
  - ☐ [VULN-27146: Container Scan: Upgrade org.apache.l \[REDACTED\]](#)
  - ☐ [VULN-25750: XRAY : Upgrade gav://org.apache.avro:avro from \[REDACTED\]](#)
  - ☐ [VULN-23700: XRAY : Upgrade gav://com.google.protobuf:protobuf-java \[REDACTED\]](#)
- md64.deb **OPEN**
- ☐ [VULN-23699: XRAY : Upgrade gav://org.apache.avro:avro from \[REDACTED\]](#)

**CLOSED**



## Future Scope

- ❑ Using agentic AI to determine if a package is exploitable or not based on historic developer feedback and open source resources.
- ❑ Improved Insights accuracy and detail
- ❑ Ability to generate and open source VEX statements alongside our product releases.
- ❑ Open sourcing goals



# Thank you!