The Security Phoenix raises from DEV-OPS ashes

From DEV-OPS Security raises in DEV-SEC-OPS-BIZ-RISK-GOV

APPSEC California 2020

PUBLIC

@FrankSEC42

https://uk.linkedin.com/in/fracipo

Agenda

About the author

Context

Evolution of DEVOPS in Security Phoenix

- Security Phoenix – Visibility Problem
- Security Phoenix – The cake and traceability problem
- Security Phoenix – People & Trust + Verify
- Security Phoenix – Scanners Triage and Visualizers
- Security Phoenix – Maturity Matrix & Education

Conclusions

Q&A

Francesco Cipollone

Founder – NSC42 LTD

I'm a CISO and a CISO Advisor, Cybersecurity Cloud Expert. Speaker, Researcher and Chair of Cloud security Alliance UK, Researcher and associate to ISC2.

I've been helping organizations define and implement cybersecurity strategies and protect their organizations against cybersecurity attacks

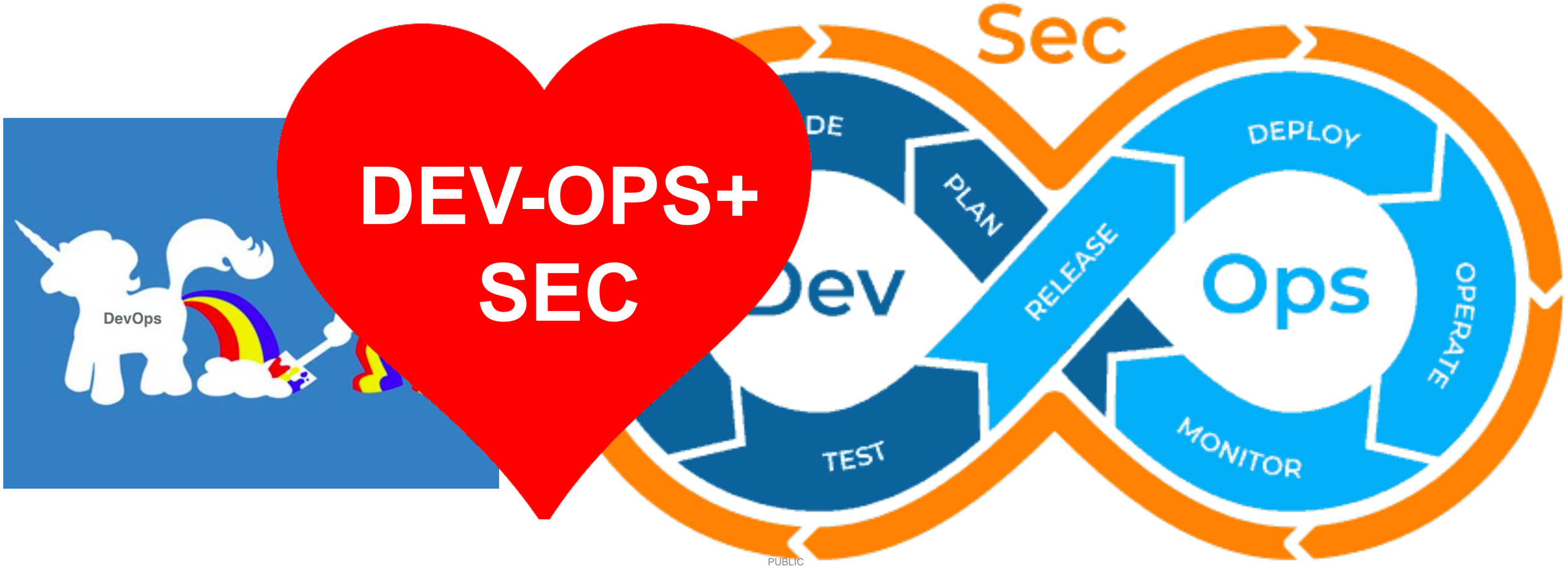
@FrankSec42 | Fracipo Linkein | Email | Website | Articles | NSC42 LinkedIn

Security is everybody's job

We need to make security cool and frictionless

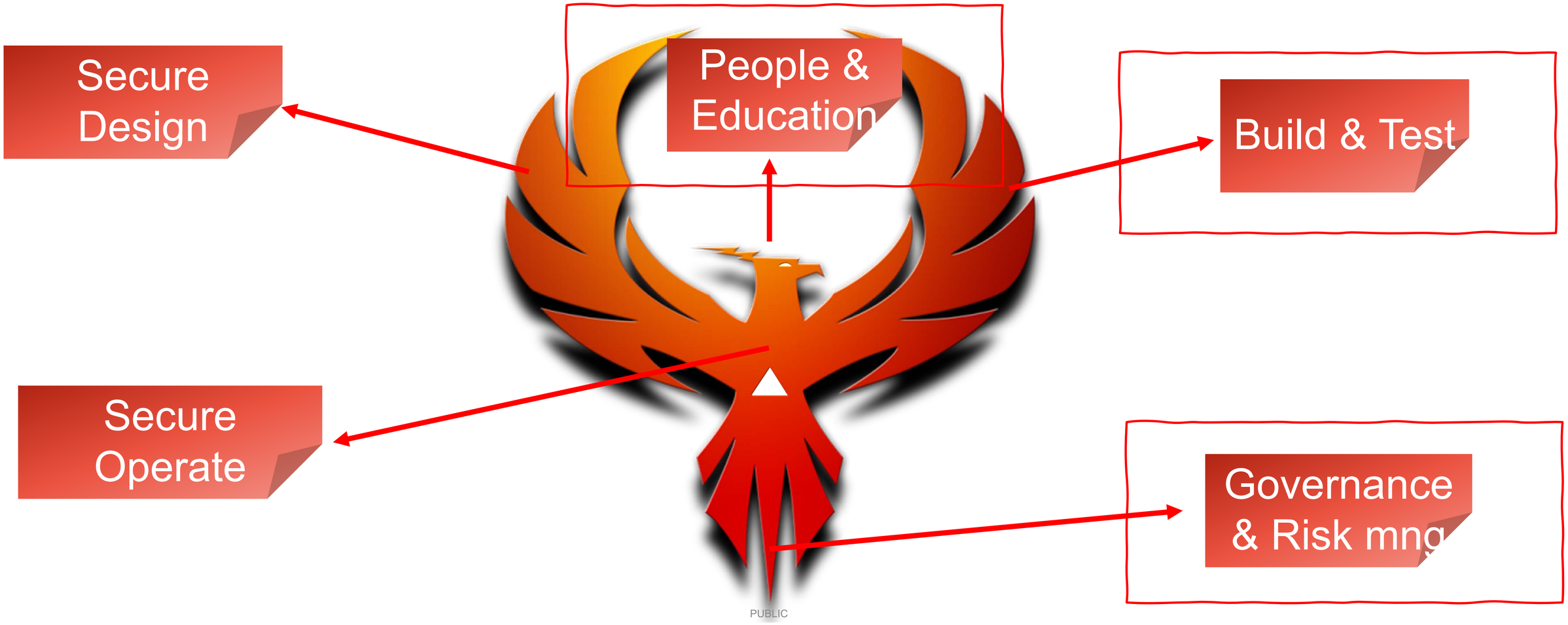
What kind of animal is the DEV-SEC-OPS?

Integrate security into the OPS team (and add a spark of BIZ)



PUBLIC

What Are the core pillars of Security Pheonix



PUBLIC

The Problem Landscape

Why do we worry about security?

Why fixing Security Vulnerabilities is everybody's job?

…because we all get affected by it

2009/2010
- Microsoft
- Heartland
- US Military
- Aol
- TJMax

2012
- Sony PSN
- NHS
- Betfair
- Steam

2012
- Dropbox
- Lastfm
- Blizzard

2013
- Yahoo(orignal)
- US Retailers
- Adobe
- UbiSoft
- Court Ventures

2014
- JP Morgan
- Home Depo
- Ebay

2015
- Deep Root
- IRS
- Anthem

2016
- Linkedin
- Friend Finder
- Dailymotion
- Mossack Fonseca

2017
- Equifax
- Myspace
- Twitter
- Yahoo

2018
- Marriot
- Twitter
- MyHeritage
- Uber
- Quora..

2019 …

PUBLIC

Major Breaches

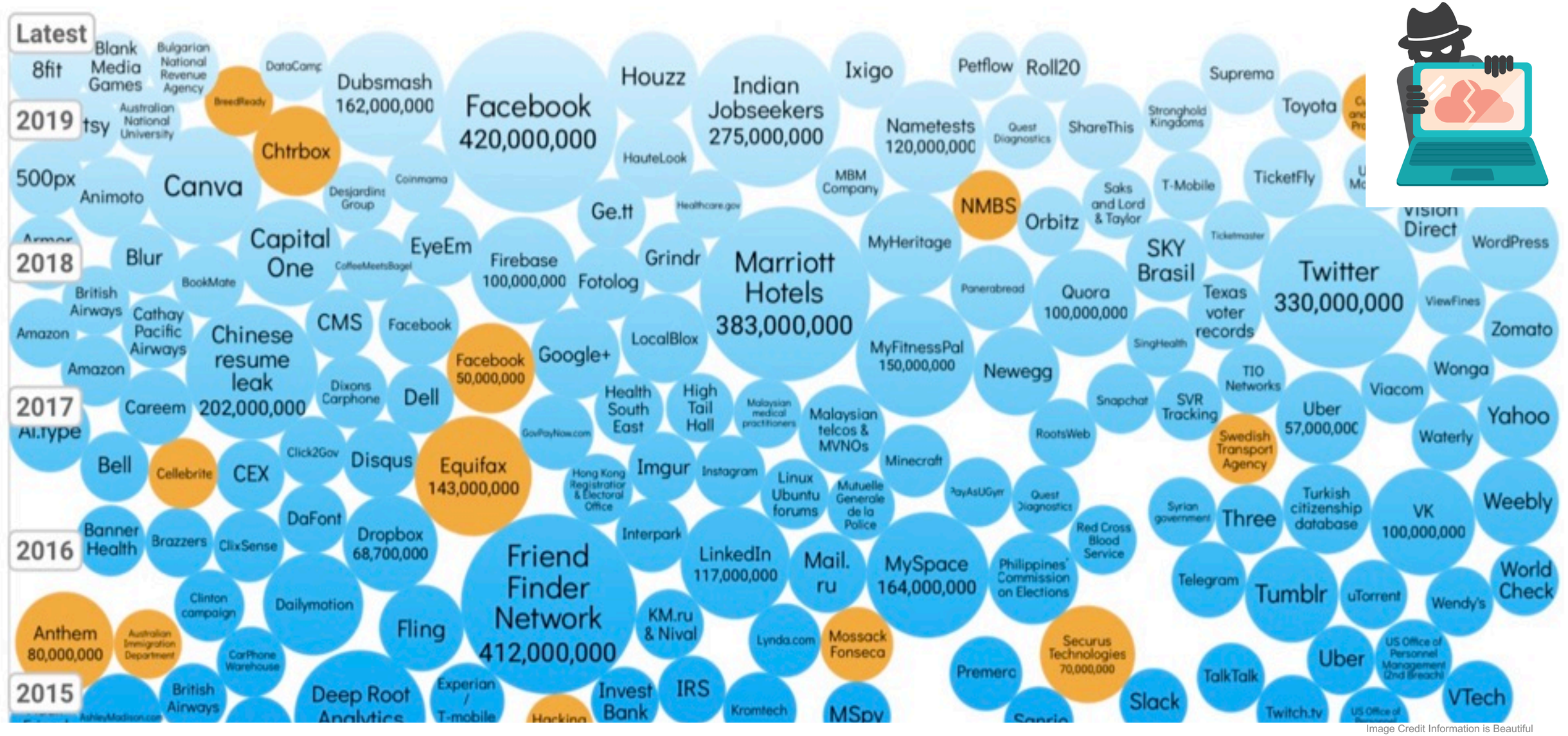


Image Credit Information is Beautiful

The Visibility Problem

From Dev to prod and cakes

PUBLIC

Understanding the room you walk in – Asset Register

To understand shine a light

Better to have full visibility

The Problem Traceability Problem

The software security cake

PUBLIC

So how we do it? Easy as baking a cake

Road to Production: the cake analogy

Design

The Objective of the various areas are

- **Ingredients**
- **Recipe**
- **Stock List (asset Register)**

Security Design

Act as Health Inspector

- **Verify Ingredients are not mouldy**
- **Verify Recipe does not contain poison**
- **Stock List (asset Register) –** verify the component used in the cake are genuine

Build/Test

The Objective of the various areas are

- **Combine ingredients (libraries+Code)**
- **Bake the cake**
- **Test the cake**

Security Build & Test

Act as Health Inspector

- **That the cake is made up of genuine ingredients (from asset register)**
- **Test the cake for mould**

Operate

The Objective of the various areas are

- **Sell The cake**
- **Restock the cake on the shelf**

Secure Operate

Act as Health Inspector

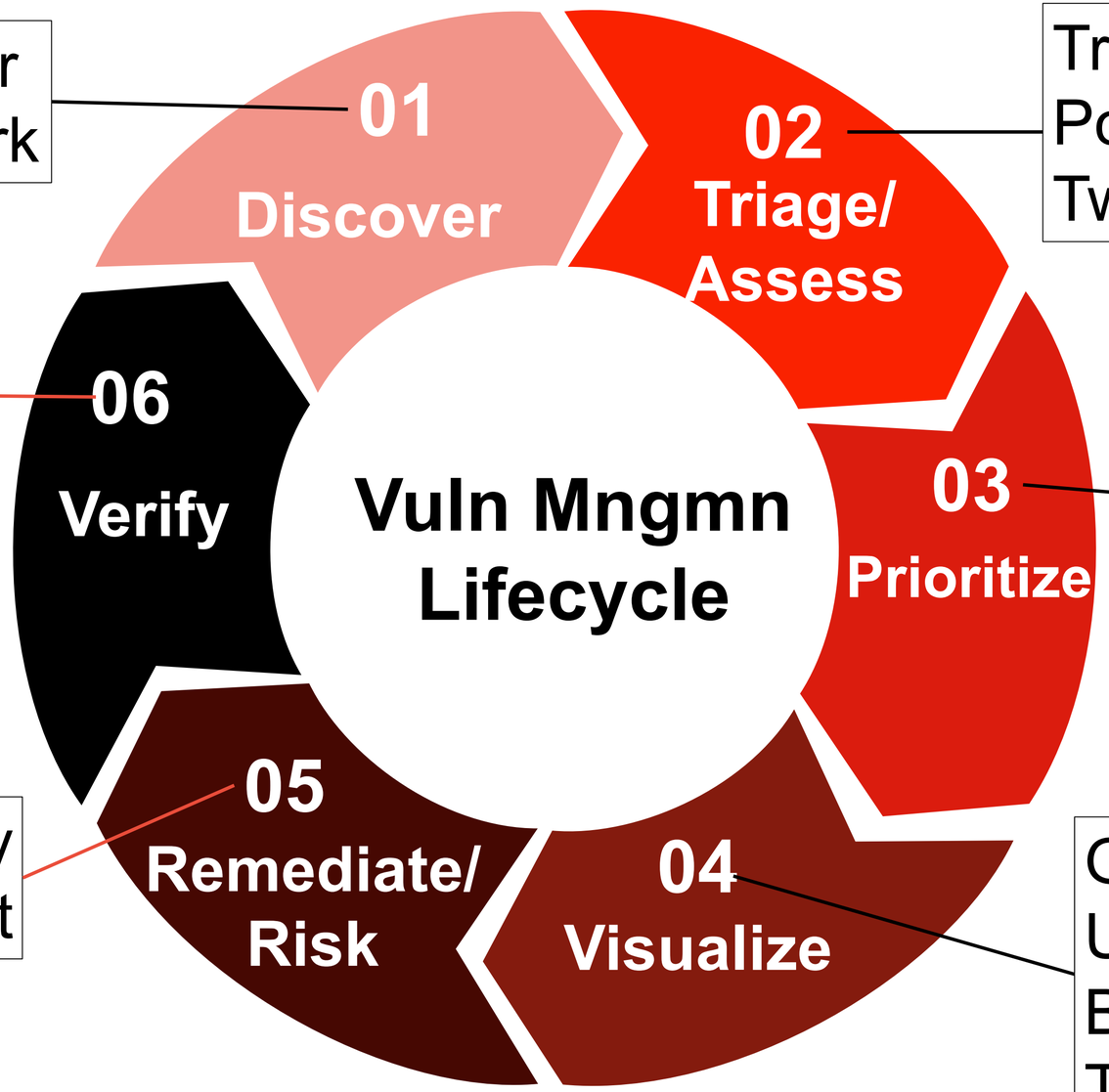
- **Verify Cake on the shop are made of genuine ingredients (from asset register)**
- **Verify expiry data of Cake**
- **Test the cake for mould**

Outcome

Asset Register for	A - IDentify (Software Asset Register)		
	Software you build (repositories)	Software You buy	Trace Completeness across all the application you have
Vulnerability Data Lake	B – Detect (Scan Code)		
	Select Team Leads and identify security champions	Get security Scanners (SAST/DAST) Onboard and teach how to triage	Create a Vulnerability Data lake (results of the vulnerabilities)
KPI Reporting & Dashboard	C – Visualize Vulnerabilities (Display)		
	Reporting Dashboards (based on the maturity & KPI)	Link the trending to Build vs FIX, Vuln trending,	
Prioritizing & Vulnerability Reduction	D – Respond/recover (Fix Vulnerability)		
	7 – Schedule Vuln Fixes (Jira)		7 – Fix Vuln & measure (quarterly)

Vulnerability management Lifecycle

Software Asset Register
Scan Code/Infra/network



Triage & identify False Positives
Tweak Scan Profiles

Prioritize vuln
(Start Easy)
Network Location
Exploitability
…

Graph –
Up/down trending of prj
Build vs FIX
Time to Fix

Fix Code and redeploy in test

Test implemented Fix

The Appsec Lifecycle & Shift Left

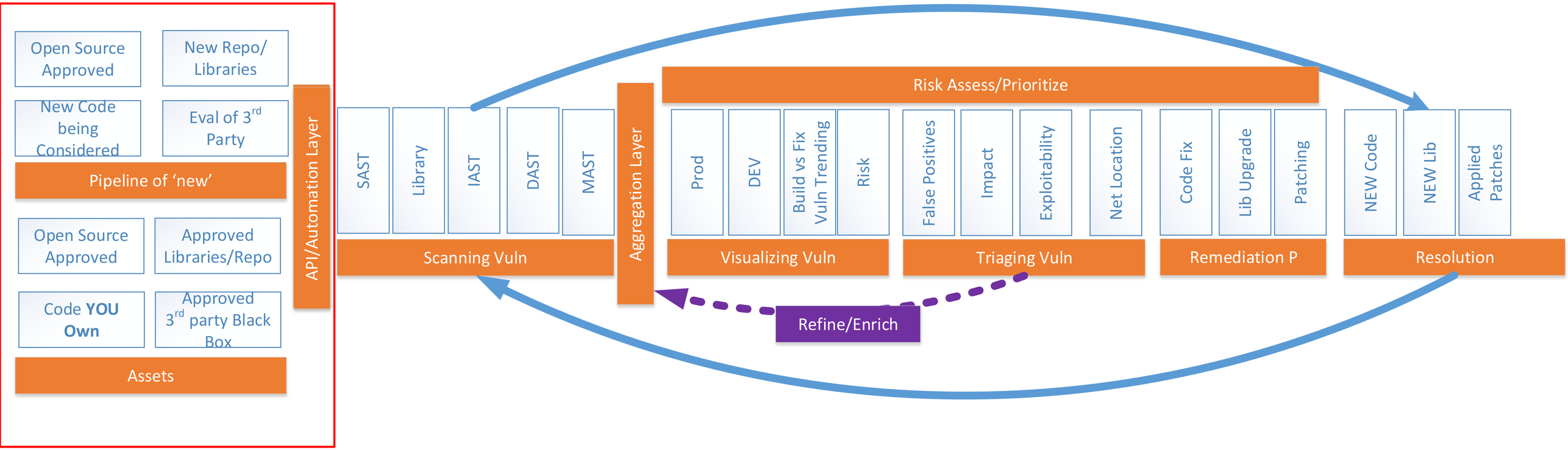
The Software Asset Register

PUBLIC

Asset Register Part A

DEV Security

Production Security



PUBLIC

Git to People: Who does what where Visualizer

DEV Security

Production Security

For Every Repo

```
git ls-tree -r -z --name-only HEAD -- update-tools-mac.sh | xargs -0 -n1 git blame \--line-porcelain HEAD |grep "^author "|sort|uniq -c|sort -nr
```

List of committers

App scan?

Build vs FIX & Tickets

JIRA

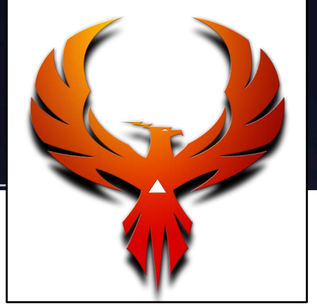
E-Mail/HR DB

PUBLIC

The People & Technology part

How to Embed DEV-SEC-OPS

PUBLIC



1. Visualize and Fix Vulnerability at scale and pace (DEV & Ops)

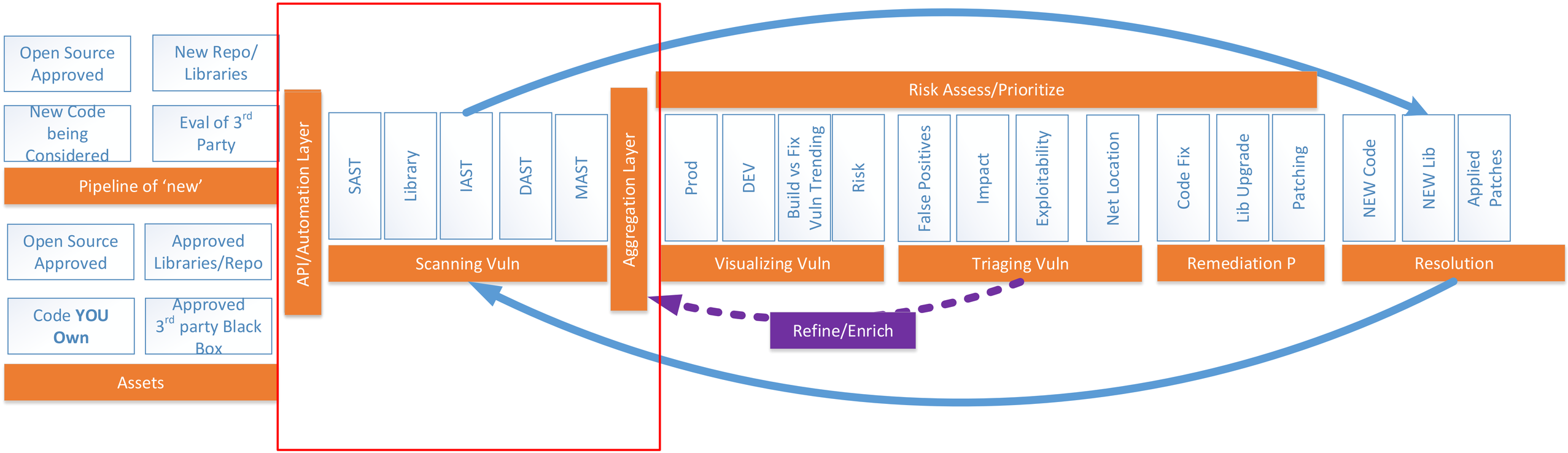
2. Trust the Product team but keep them accountable: Trust & Verify & License to Operate

3. Maturity & Recap

Dashboard for Code Defects – Part B

DEV Security

Production Security

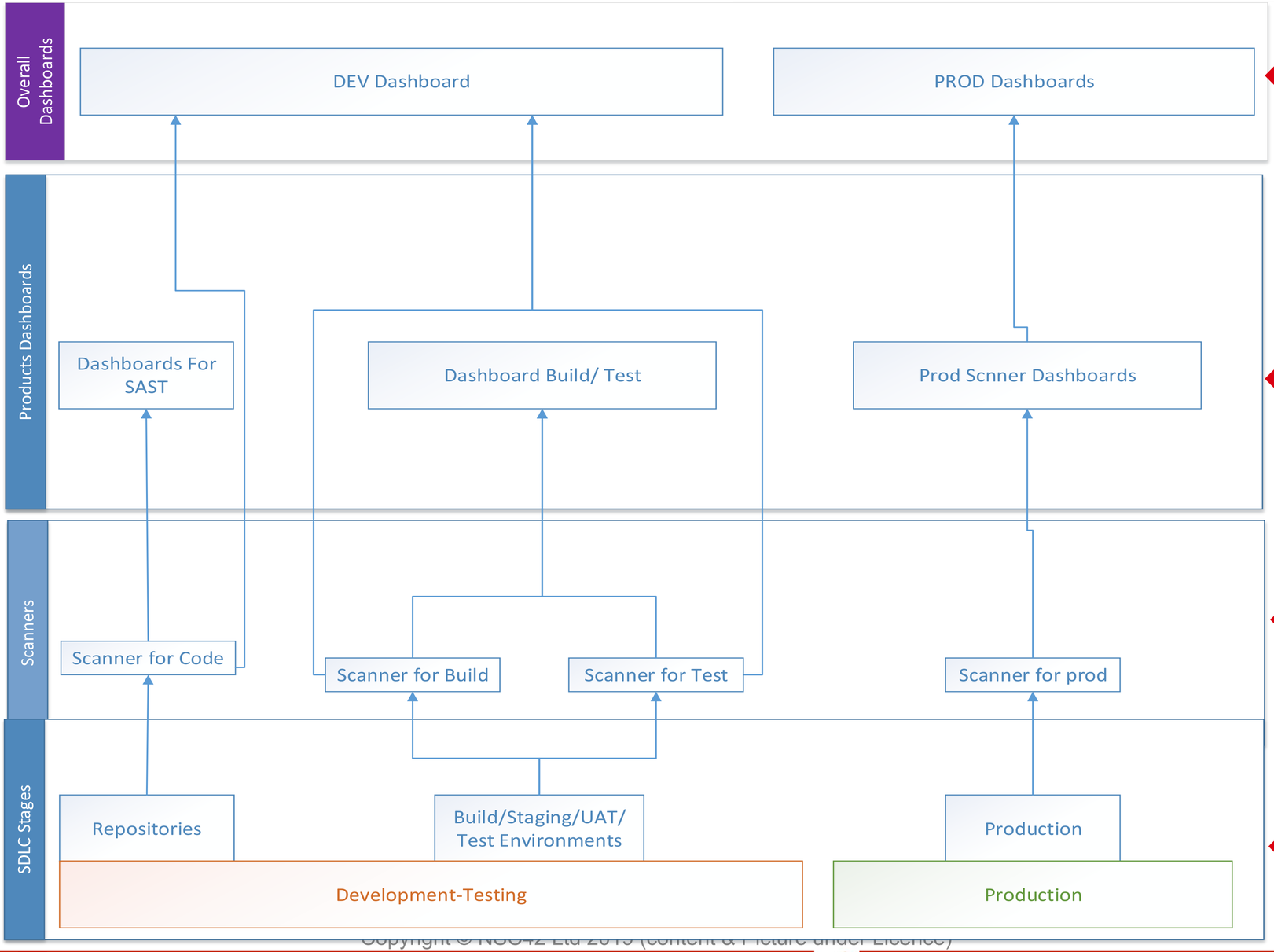


PUBLIC

Dashboard for Code Defects -> Under the hood

DEV Security

Production Security



SET Targets For Prod & DEV Vuln

Triage the vulnerabilities

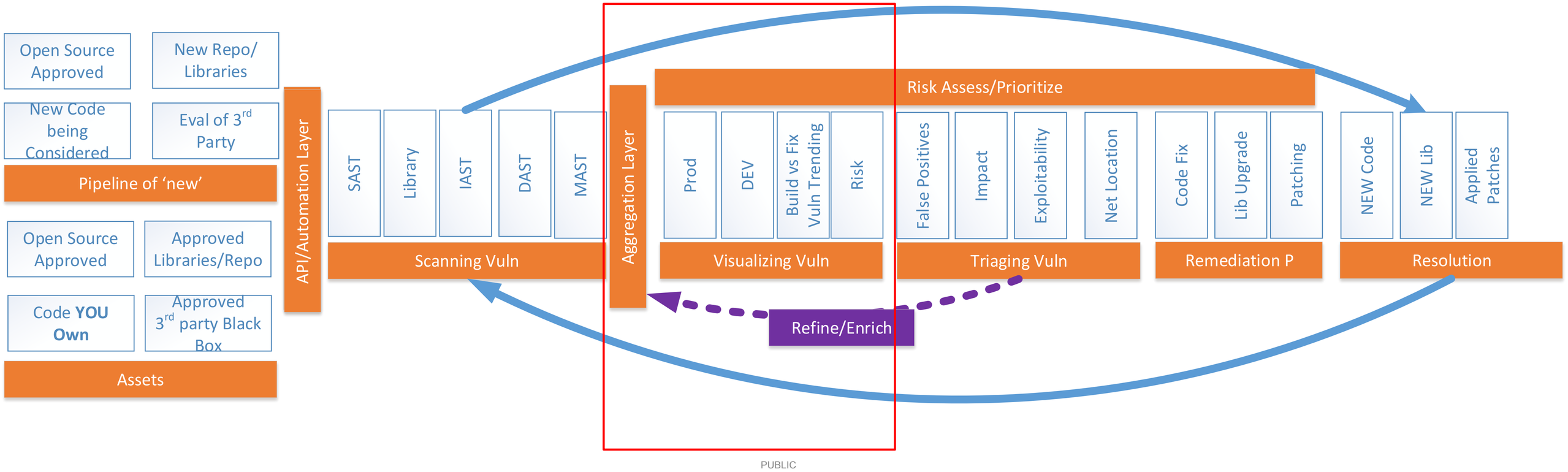
Scanners to Tickets or aggregators

Scan At various Stages

Dashboard for Code Defects – Part C

DEV Security

Production Security

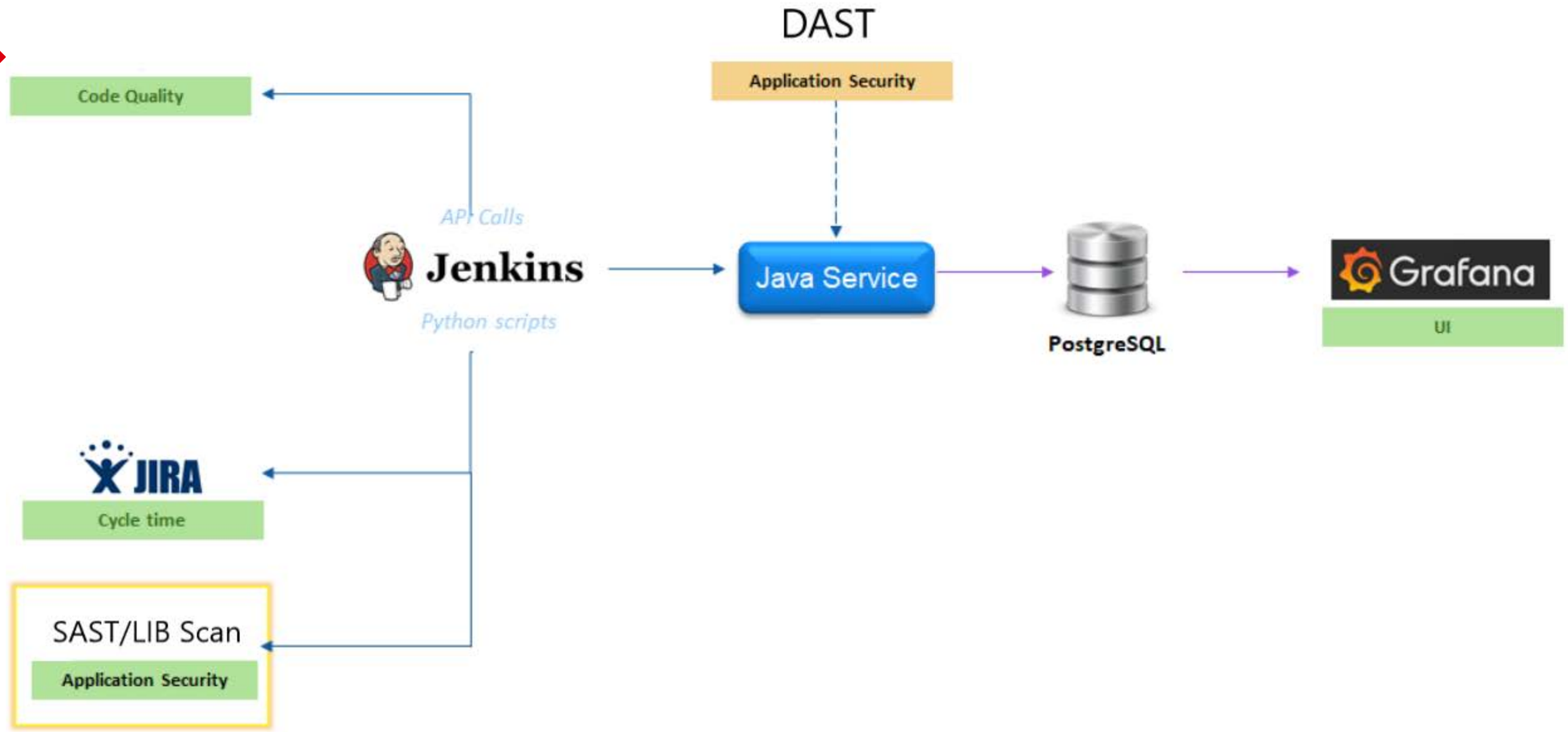


PUBLIC

Dashboard for Code Defects

DEV Security

Production Security

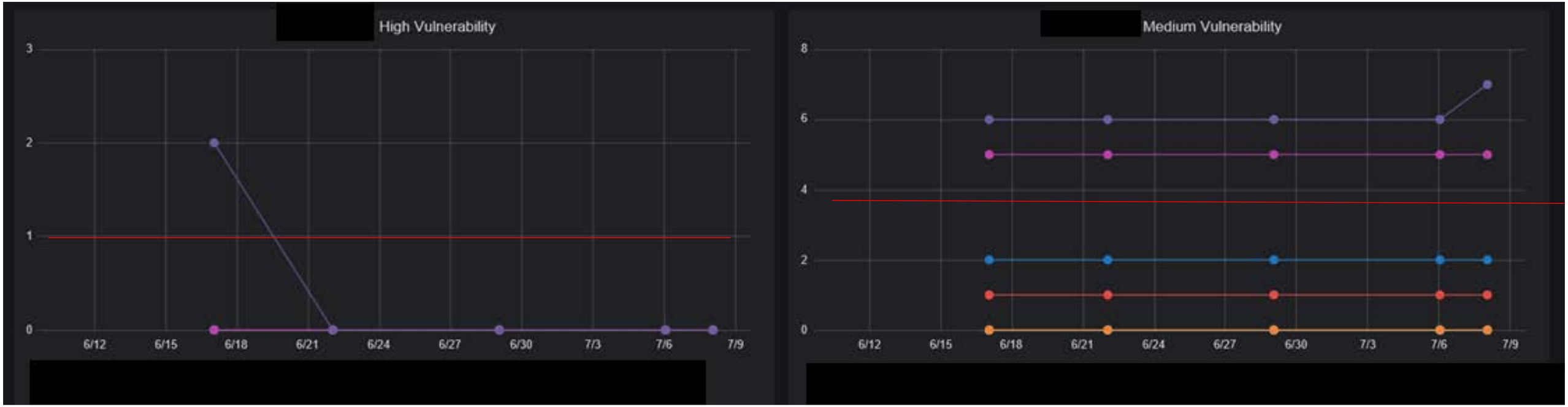


PUBLIC

DEV Security

Production Security

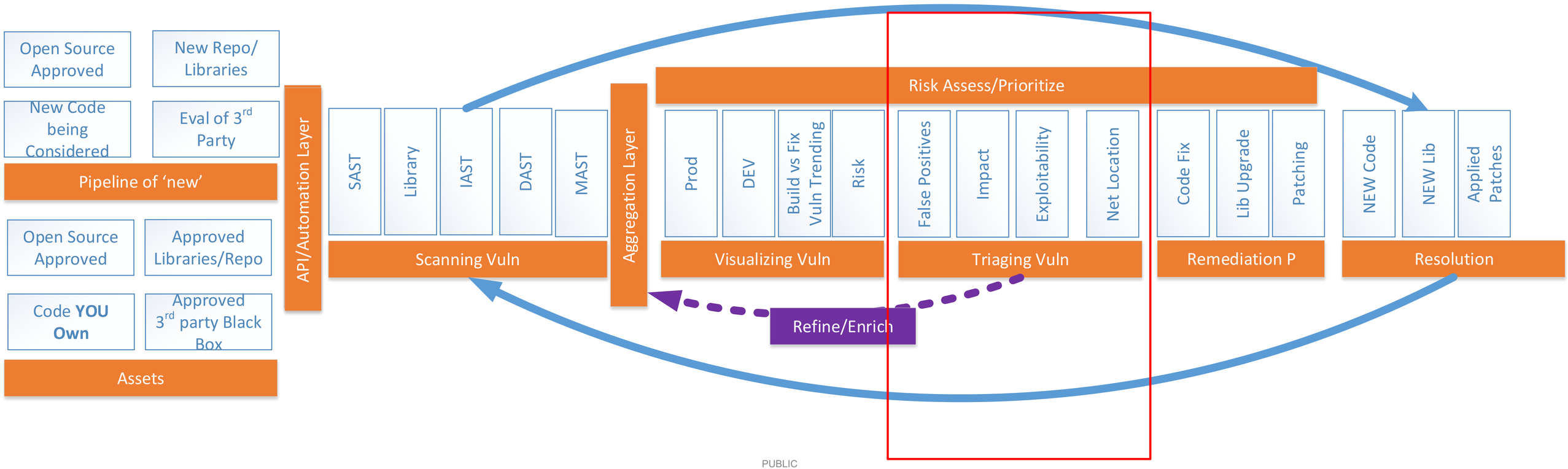
Example of a dashboard for Vulnerability Visualization



Dashboard for Code Defects - Part D fix

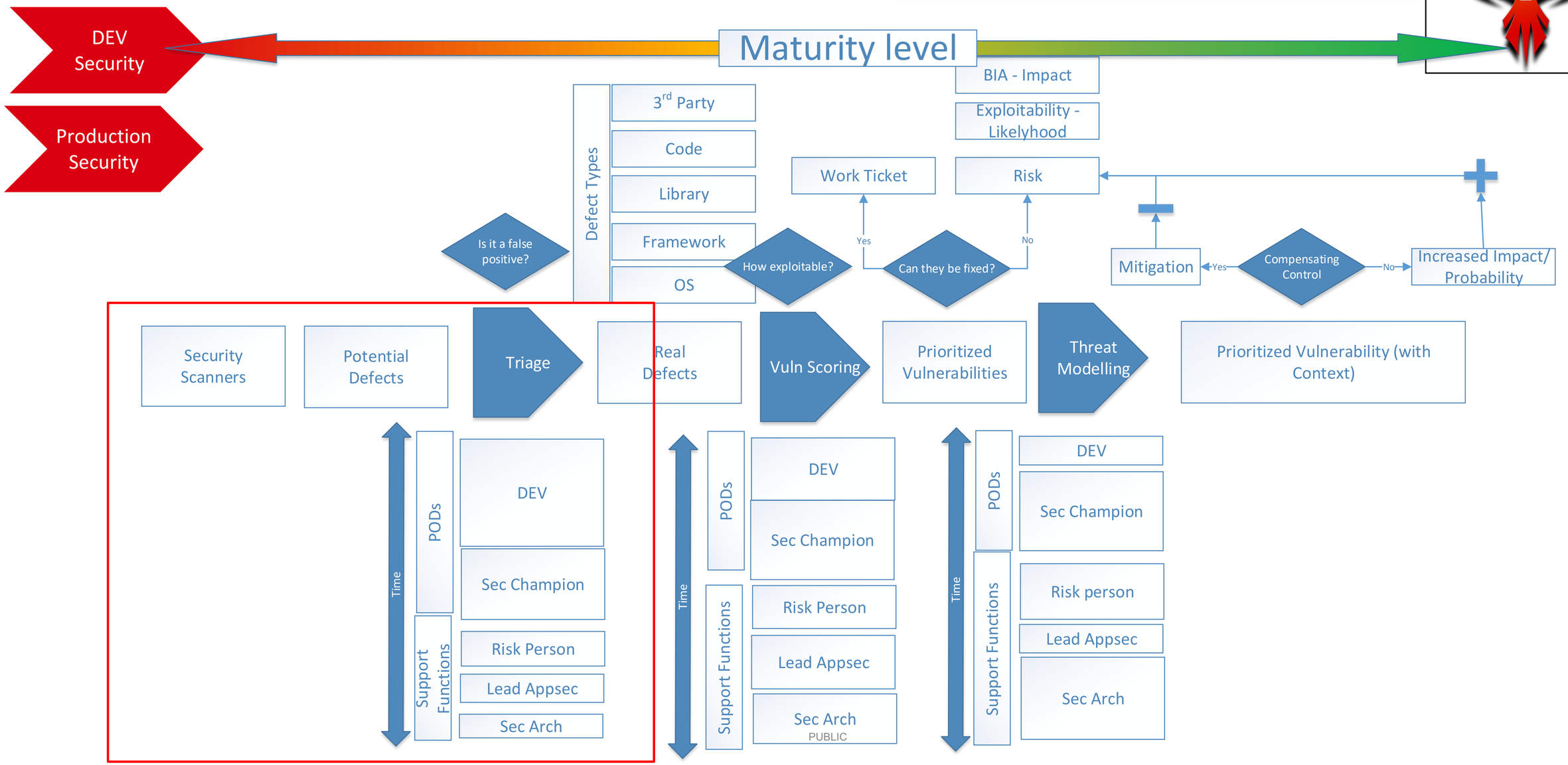
DEV Security

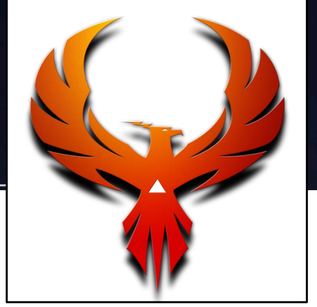
Production Security



PUBLIC

Triaging Vulnerabilities





1. Visualize and Fix Vulnerability at scale and pace (DEV & Ops)

2. Trust the Product team but keep them accountable: Trust & Verify & License to Operate

3. Maturity & Recap

Going fast but with confidence (SEC)

1. Trust & Verify

2. License to operate/code

People & Education

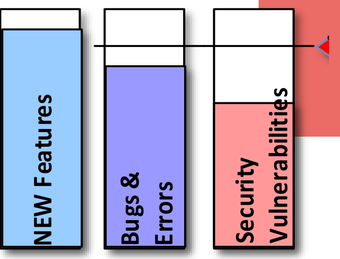
>> Set Thresholds: Bild vs Fix, Vulnerability trending

PUBLIC

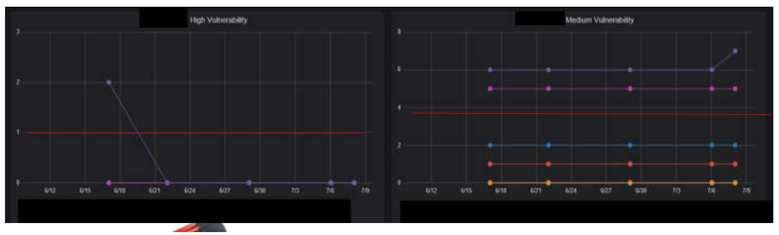
Trust & Verify – Under the hood

Learning & Education

Build vs FIX Target



Vulnerability Targets (Quarter)



Phoenix Aggregator DB

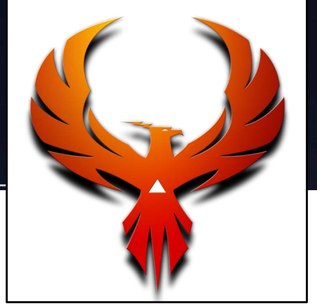
License to operate

PUBLIC

Developer can operate fast and deploy if they have a license

1. Trust your developers and apply a ‘license to operate’
2. Apply governance (light and heavy weight)
3. Visualize and keep everyone accountable
4. Make security resource available to the developers and document the fixes

PUBLIC



Security Education

Education:

1. Awareness Training For your users

2. Craft Training based on the scanner (faults) data

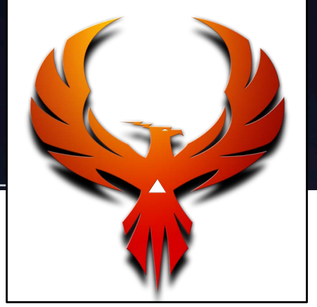
3. Education on the job – What good looks like

4. Make the training entertaining (CTF and Rewards)

Maturity Model & Recap

Bringing all together

PUBLIC



1. Visualize and Fix Vulnerability at scale and pace (DEV & Ops)

2. Trust the Product team but keep them accountable: Trust & Verify & License to Operate

3. Maturity & Recap

To Achieve High Maturity what do you do

DEV Security

Production Security

Overall Maturity

	Level 1	Level 2	Level 3	Level 4	Level 5
	Intial	Managed	Defined	Quantitatively Managed	Optimized
Security Design	AS-IS->TO-BE				
Security Design Governance	AS-IS	TO-BE			
Security Build & Test	AS-IS	TO-BE			
Security Operate		AS-IS-> TO-BE			
Appsec Security Education	AS-IS	TO-BE			
Application Security Risk Management	AS-IS	TO-BE			

Maturity Steps

Maturity	Task	How much
1	No Peer Review	
1	No Code Scanning	
1	No library update	
1	No risk management	
1	No visibility on vulnerabilities	
1	No knowledge of pods	
1	No team to pod mapping	
1	No Documentation of Fixes	
2	Peer Revew	
2	Team to Pod to Stash recorded	
2	Onboarded application on code scanners	
2	Libray scanning	
2	triage of vulnerabilities (base) - Consider only high medium and low	
2	Manual Evaluation of team and Allocation of Licence to Operate	
2	No SLA	
2	No Documentaiton	
2	Adoption Dashboard	
3	Peer Review with Toolset	
3	Updated Teams and asset register	
3	Basic Triage of vulnerabilities	
3	**Code Scan with Pipeline Break & Basic SLA**	
3	Adoption Dashboard (advanced) Per A.C. and Per Region	
3	Risk Assessment from code scanning with record in Risk management Register	
3	Automated Licence to operate: Code Scanning, Libraries, Internal Training	
4	Fix time of vulnerabilities recorded	
4	T-Shirt Sizing of fixes and Adaptation of SLA based on fixes	
4	Visualization of pod to fix	
4	segmentation dev and prod	
4	Fix ticket in Jira & Build vs Fix Concept	
4	Fuzzing (basic with generic per app)	
5	Automated Licence to operate: Code Scanning, Libraries, Internal Training, Build Vs Fix	
5	Automated Fuzzing & Library of tests	

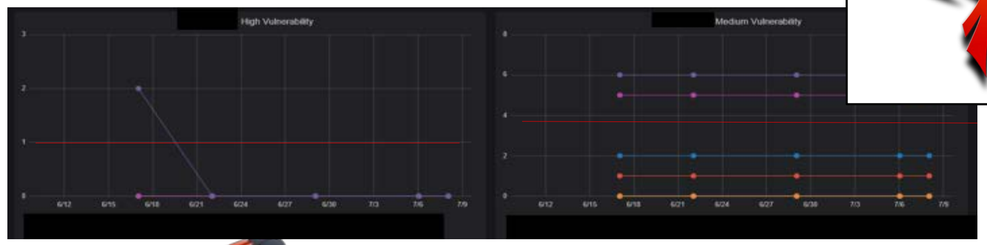
KCI

			Reporting Frequency
	KCI Maturiy	Build/Test	
Prereq ->	0	Who is working on which repository	Monthly
	1	Team On-boarded on scanners (per pod)	Monthly
	1	Code Scanning Frequency per project (min 1 per week)	Monthly
	1	Dashboard for Scanners created	Monthly
	2	Number of vulnerabilities ticket recorded	Weekly
	2	Dashboard for vulnerabilities - Onboarded Projects	Weekly
	2	Vulnerability Fixed (quarter)	Monthly/ Quarterly Checks
	3	Project imported in Kennar and Enriched Vulnerabilities (kennar)	Monthly
	3	Projects breaching the Build vs Fix target	Monthly/ Quarter Checks
	4	Fixes per thematic in SLA	Monthly
	4	SLA for Fixes (breached/achieved)	Monthly
	4	Team Achieving Licence to operate and Out of the licence	Monthly
	5	Build vs fix	Monthly
	5	Licence to operate	Monthly

Outcome

Asset Register for	A - IDentify (Software Asset Register)		
	Software you build (repositories)	Software You buy	Trace Completeness across all the application you have
Vulnerability Data Lake	B – Detect (Scan Code)		
	Select Team Leads and identify security champions	Get security Scanners (SAST/DAST) Onboard and teach how to triage	Create a Vulnerability Data lake (results of the vulnerabilities)
KPI Reporting & Dashboard	C – Visualize Vulnerabilities (Display)		
	Reporting Dashboards (based on the maturity & KPI)	Link the trending to Build vs FIX, Vuln trending,	
Prioritizing & Vulnerability Reduction	D – Respond/recover (Fix Vulnerability)		
	7 – Schedule Vuln Fixes (Jira)	7 – Fix Vuln & measure (quarterly)	

Learning & Education



Deployment to prod
Relies on the License to Operate

Code
3rd parties Components (FOSS + Libraries)
Application

Prod
Test
DEV

Production Dashboard
Development Dashboard

Am I compliant with
Code Defects Target ?

Security Scanners

Defects

Thresholds

NEW Features
Bugs & Errors
Security Vulnerabilities

Job Queue

Bugs

New Features

Am i still compliant with Overall
Build vs FIX Targets ?

Application/ Product Owner
Security Champion
Security Architect
Engeneers & Developers
Engeneers & Developers

Triage & Vulnerability Per applicationDay to day fix or build

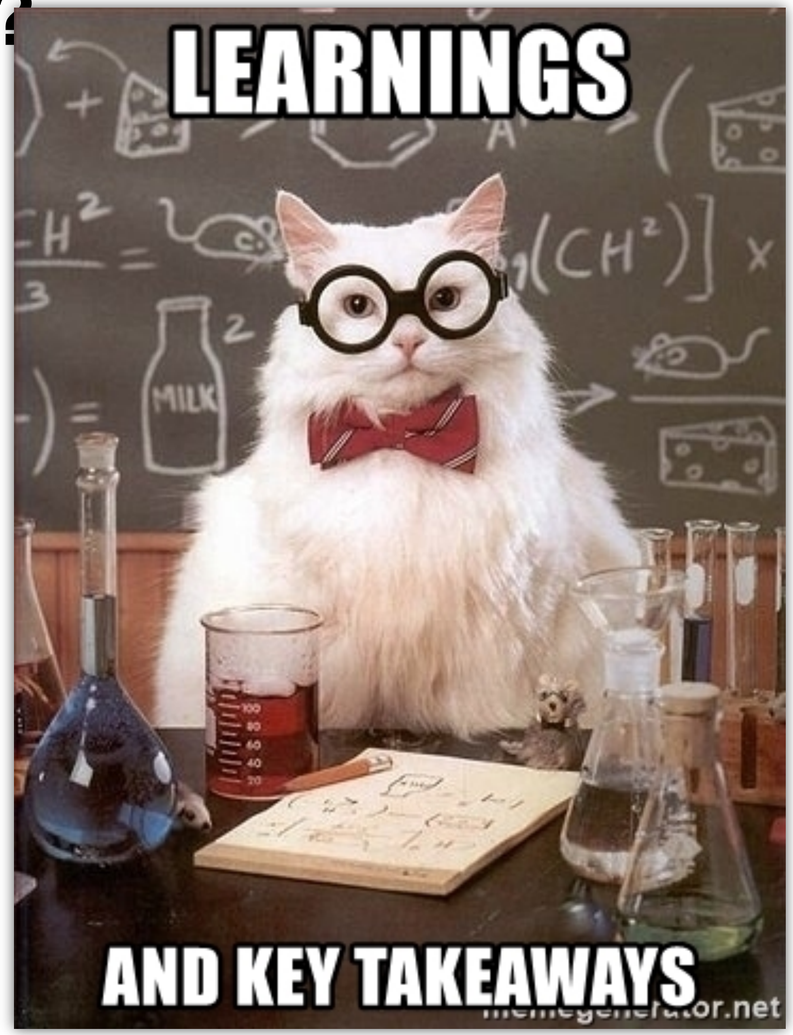
DEV-SEC-OPS Application Group (unit that works on one or more application)

PUBLIC

Security is everybody's job

Does develops have a single solution?

- Trust And Verify
- Vulnerability Management every day life
- Automation vs people aspect – is a transformation
- Data Driven Education
- Governance at scale



Cyber #MentoringMonday Podcast

Every 2 weeks 1.30 PM UK Time

slack

PUBLIC

cloud security alliance SM UK & IRELAND

NSC42

@FrankSEC42

(ISC)² OFFICIAL CHAPTER LONDON

Cyber Security Awards

cloud security alliance SM
CSA UNITED KINGDOM

Cyber Security Awards 2020

Cloud Security Influencer of the Year

Submission – 10 of May 2020 (TBD)

Ceremony 4 July 2020

#CYSECAWARDS20

https://cybersecurityawards.com/

https://cloudsecurityalliance.org.uk

Submit: info@cybersecurityawards.com

Info: Francesco.Cipollone@cloudsecurityalliance.org.uk

Q&A

NSC42

WHEN YOU ARE CYBERSAFE WE ARE CYBERHAPPY

Thank you

Get in touch:

@FrankSEC42

https://uk.linkedin.com/in/fracipo

Francesco.cipollone (at) nsc42.co.uk

www.nsc42.co.uk

PUBLIC