JFROG CURATION
SEAMLESSLY CURATE SOFTWARE PACKAGES ENTERING YOUR ORGANIZATION

Bill Manning, Solutions Engineering Manager

@williammanning
SOFTWARE SUPPLY CHAIN SECURITY

- Life With JFrog Curation
- Should We Shift Further Left?
- Rolling out Curation in JFrog
- Curation New & Upcoming Features
Life with JFROG CURATION
LIFE WITH JFROG CURATION

- **Centralize Visibility and Control** of 3rd party (OSS) package downloads

- **Automate Curation of 3rd Party Packages** to provide your developers with a trusted source of software components

- **Improve DevSecOps Experience** & Realize Cost Savings with seamless integration and reduced remediation later in your SDLC
LIFE WITH JFROG CURATION
LIFE WITH JFROG CURATION

First Line of Defense for any SDLC Program
Should We SHIFT FURTHER LEFT?
SHOULD WE SHIFT FURTHER LEFT?

MALICIOUS PACKAGES

- Protect Any Pipeline Automatically
  - SDLC Program Maturity Agnostic
  - Pipelines Under The Radar
- Optionally Protect Developer Local Env

Block Before Use

NPM malicious package reports

- 8000
- 6000
- 4000
- 2000
- 0
- 2020H1 2020H2 2021H1 2021H2 2022H1 2022H2 2023H1
DEMO PART I
Blocking malicious
Demo Part I (3 minutes) - Block malicious

**Script**

- preset:
  - ensure no block malicious exist

- open Curation webapp

- Present the curated remotes and show no malicious package coverage in the organization

- Add block malicious packages on all org policy

- Present the coverage in curated repositories page

- open vs code in cors demo folder

- run npm install in terminal without cors.js -> success

- add cors.js latest and fail

- present the audit of it as failed

Explain the ease of protecting from malicious across the org from now forward with this policy
SHOULD WE SHIFT FURTHER LEFT?

FORCING FIX (UPGRADE) WHEN AVAILABLE

Critical / High CVEs With Fix

- 83% Maven
- 63% npm
- 89% pypi

- Avoid later stage risk
- Covers any pipeline
- Different SDLC Maturity Stages
DEMO PART II
Forcing fix (upgrade) when available
Demo Part I (4 minutes) - Fix fixable

Script

○ preset:
  ■ block on critical with fix only

○ open in vs code the folder “electron demo”

○ make sure there is a policy with “block critical with a fix”

○ npm install in the terminal and `open-graph 0.2.6 (Latest)`

○ run ./jf ca and present the failure in tough-cookie transitive

○ present the open-graph in catalog and the tough-cookie transitive critical with a fix

○ explain we need to use overrides maybe

○ add override to tough-cookie latest version

○ run npm install

○ run ./jf ca

○ explain the fix at the gate values and mind set used
SHOULD WE SHIFT FURTHER LEFT?
FORCING REPLACEMENT WHEN NEEDED

Packages Added in 2022

15%  44%  22%

Maven  npm  pypi
SHOULD WE SHIFT FURTHER LEFT?
FORCING REPLACEMENT WHEN NEEDED

“Unmaintained” Packages
(latest older than 2 years)

53%  42%  50%

Maven  npm  pypi
SHOULD WE SHIFT FURTHER LEFT?
FORCING REPLACEMENT WHEN NEEDED

Critical / High CVEs No Fix

17%  37%  11%
SHOULD WE SHIFT FURTHER LEFT?
FORCING REPLACEMENT WHEN NEEDED

Unmaintained with Critical / High CVEs No Fix

33% 65% 42%

Maven npm pypi
SHOULD WE SHIFT FURTHER LEFT?
FORCING REPLACEMENT WHEN NEEDED

Unmaintained Package  High Operational & Security Risk
SHOULD WE SHIFT FURTHER LEFT?
FORCING REPLACEMENT WHEN NEEDED

Cost of Replacement

Minutes ➔ Hours ➔ Days ➔ Weeks

PREVENT ➔ DETECT

CURATE ➔ CREATE ➔ PACKAGE ➔ PROMOTE
DEMO PART III
Forcing replacement when needed
Demo Part III (4 minutes) - Blocking if no fix and waiver

Script

- preset
  - block critical with or without only
- open in vs code jxon project
- present the jxon version used
- open the terminal and run npm install
- present the failure
- run ./jf ca and present the block on transitive xmlldom
- present the data in Catalog for jxon and xmlldom
- explain the option to waive because of popularity
- explain the model of force remediation at the gate
ROLLING CURATION IN JFROG

~500 Developers

US, India, Israel, Ukraine, France

Globally Distributed

Curated Stack

Maven

npm
ROLLING OUT JFROG CURATION IN JFROG
HOW WE ARE DOING IT

Phase I: Dry-run & Block Malicious

- ~11K distinct assessed
- Critical/High CVEs (170 fix / 14 no fix)
- Unmaintained (1.5k npm / 16 maven / pypi 36)

Next:

- Curate remaining eco systems
- More Blocking Policies as
  - Critical with fix
  - Unmaintained
  - Banned licenses
SOME NEW FEATURES

Audit API
CURATION AUDIT API

https://jfrog.com/help/r/jfrog-rest-apis/jfrog-curation-rest-apis

- Extract package blocked/approved events
- Send to log management systems
- Send to SIEM/SOC threat detection systems

Approved/Blocked Audit

Get Approved/Blocked Audit Logs API

Description: An API to get Approved/Blocked audit events for packages (allowing full export).
Note: A synchronous REST API, to get Approved/Blocked audit events for packages.
Since: 3.82.x
Security: Requires a valid user with the "VIEW_POLICIES" permission.
Usage: GET /xray/api/v1/curation/audit/packages
DEMO PART IV

New features
Demo Part VI - Dashboard & Docker Hub (5 minutes)

Script

○ present the new dashboard

○ present the new docker conditions and policies in place for the docker remote (effective view)

○ present blocking of a package that is both malicious and not docker official

○ present the image
  https://hub.docker.com/r/docker2021repos/postgres

○ try to pull it using:
  docker pull swampupdemo01.jfrog.io/docker-remote-repo/docker2021repos/postgres

○ present the failure in the audit

○ present the malicious in the dashboard
UPCOMING NEXT...

- Policies for group of remote (JFrog Project)
- Customized conditions
- Cross JFS policies
- Additional eco systems
- Support Curation in JFrog IDE plugin
- Waiver management (APIs, ServiceNow)
- Manage all via APIs
- Additional notification channels
QUESTIONS?
THANK YOU!