Hills Draft Review, GitSecure

10/15/2020

Problem Statement:
How might we automate secure dependency management that will provide the development team with the notifications and insights needed to address insecure dependencies on IBM Cloud?

Who (our users)	What (pain-points/goals)	Wow (grounded in time, money, or efficiency)
Chief Security Officers	Establish policies for application security	Reduce time to identify noncompliant dependency by XX%
	Notify users earlier about issues	Reduce number of engagements with developers by XX%
IBM Developers	Developers only see alerts after code is pushed to production (shift process left)	Reduce time to identify possible dependency updates from hours to seconds
	Developers don't have visibility into correct dependency versions to use	Reduce developer workload when fixing vulnerabilities from 10% of their time to 1%
	Developers aren't aware of dependency update prioritization	

Jim Doran Key Takeaways

Recording Link: https://ibm.webex.com/ibm/ldr.php?RCID=860f28a3e967471fb52878bd0dab0fc3
Password: 3Gpw2Uvv

Hills

- Measure defects in image build (old) vs defects in pull request (git-secure)
- Reducing time to fix vulnerabilities from 10% to 1% is a good hill
- If we don't have measurements -> Run through experiment (e.x. old Redis version) to see how long to fix

Other feedback
- KPI graphs/factoids for GitSecure in final presentation
- Organization level view of dependency security

Notes (Oct 19)

Visibility and Control - Hallmarks
Supply chain of software with 3rd party dependencies
Modern processes opaque
Old vs git-secure
- CI/CD push -> Vulnerability scan/hygiene report (already pushed)
  - Time consuming in registry
- Finding vulnerability in git vs downstream
  - Questions are often asked downstream
    - **What git repo?**
    - **What developer?**
-For a given set of applications
  - **How many defects in image level vs number of defects in pr**
  - See what vulnerabilities were never detected
-Measuring performance?
  - Tunable parameters optimization
- Notice vulnerability found
  - **Time to fix vulnerability (hard to find person too)**
  - Example for difficult remediations
    - Test out with application
- Proactive
  - Treat intel, CVE, new Python package
  - **Even if no PR - we need to do scanning on existing deployments**
  - Trust/risk aversion
    - Test with application

Developers not experts in security
- Confirm/deny remediation

Value
- 150000 GitHub repos in aggregate

    - Trends of CVE detection
    - Effect of CVE across enterprise
        - X number of repos
    - Smart remediation
- Dependency graph model analytics
    - Anomalies
    - Trends
    - Impacts
- "heres the insight we found" in graphs
    - **Metrics and factiods around git-secure**
    - **KPIs**
    - Does this exist in Github?
    - What is scope of graph
        - What could you learn
        - Enterprise level visibility
        - Might have to make the query yourself
        - How are we doing as an org?
        - Does lack compliance assessment, only code level
        - Building on top of GitHub
        - Complimentary data source + hints