Microsoft Azure

Secure DevOps: Application Security Principles and Practices

Automating a Secure and Compliant pipeline

Your name
Your Title
Microsoft

Module Overview

- Shift left testing
- Managing secrets
- Lab – automating a secure pipeline

Microsoft Azure

Shift left testing

Static Application Testing

Source code scanning should be covered by implementing Static Application Security Testing (SAST)

Automate SAST in the pipeline

Segment SAST for different builds

- CI builds
- PR Security Verification Tests (SVT) builds

SAST should be a requirement for commits to the main branch

Security in code reviews

Start with SAST

- Coding errors
- Credential scanning
- Security issues

Search for

- Home grown crypto or outdated hashing functions
- Bad use of API wrappers
- Utilization of bad utility classes
- Improper logging techniques

Dynamic Application Security Testing

Dynamic Application Scanning Tools (DAST) are designed to scan the staging and production website in running state, analyze input fields, forms, and numerous aspects of the application against vulnerabilities

Secure your cloud environment deployments

Perform Security Verification Tests on your configurations before deploying

Best Practice is to leverage Infrastructure-as-Code (IaC) concepts and rebuild when possible

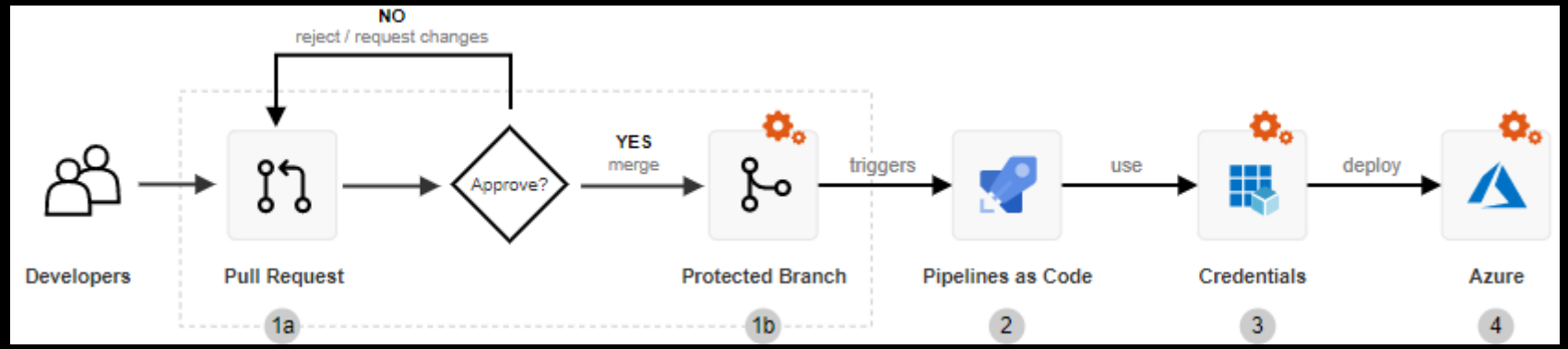
Utilize Azure policies and the Secure DevOps Kit for Azure for enforcement and configuration drift

Secure your pipelines

Deployments to production should be fully automated

Only DevOps Engineers (limited team) should have access to edit the Prod pipeline

Monitor activity by leveraging the Azure DevOps Organization Auditing Feature



Use a phased approach

The complexities of taking a Secure DevOps approach can be lessened by using a tactical and phased deployment

Focus on quick wins or “low hanging fruit”

Once you reach a specific state, move on to the next phase

Evaluate which portions can be shifted and monitored in the developer’s IDE

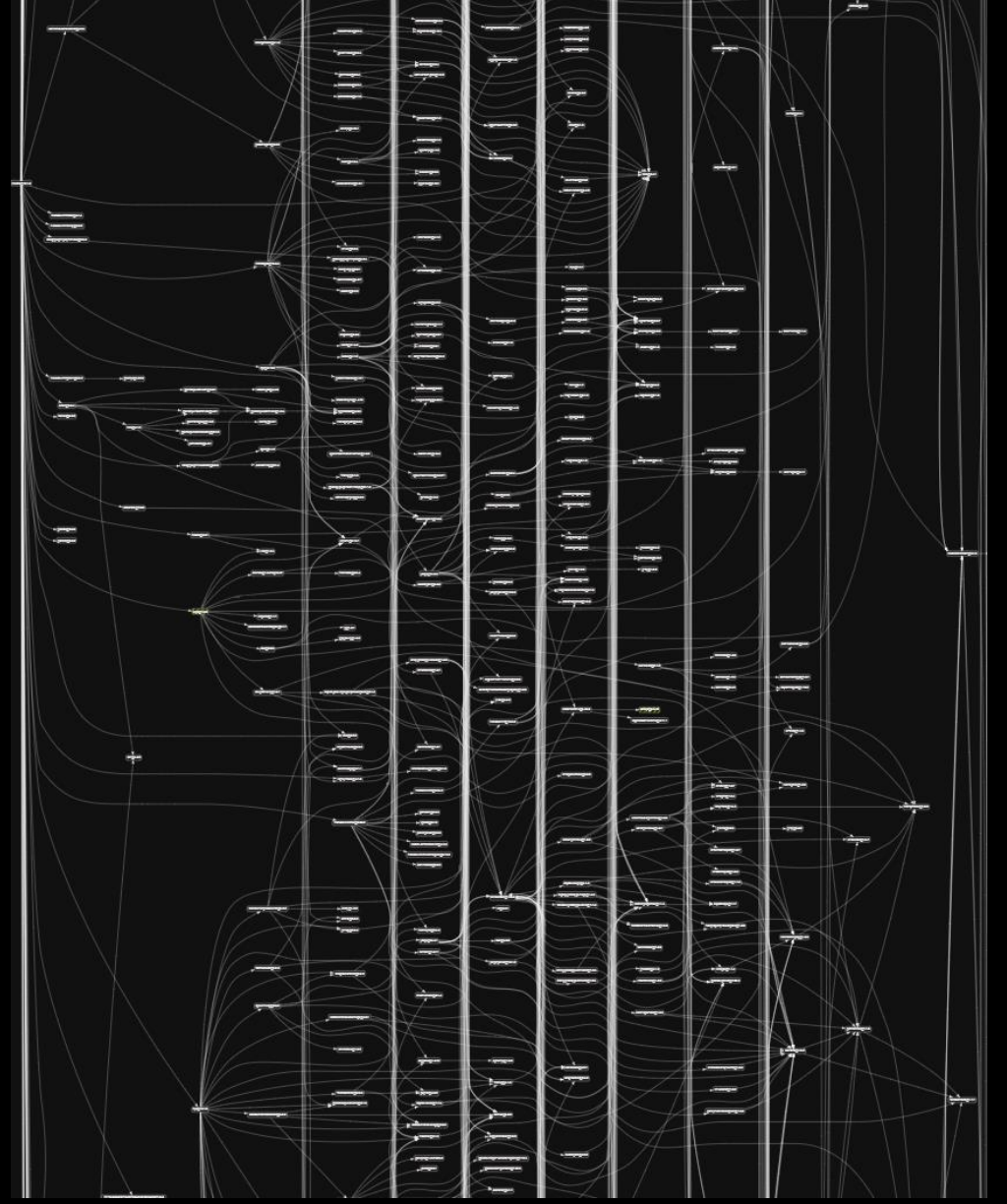
- Run credential scanning in the IDE
- Just in time learning with Security Intellisense in the Secure DevOps Kit for Azure
- Incorporate a linter in the developer IDE
- Require developers to run Static Code Analyzers in the IDE

Microsoft Azure

Software Composition Analysis (SCA)

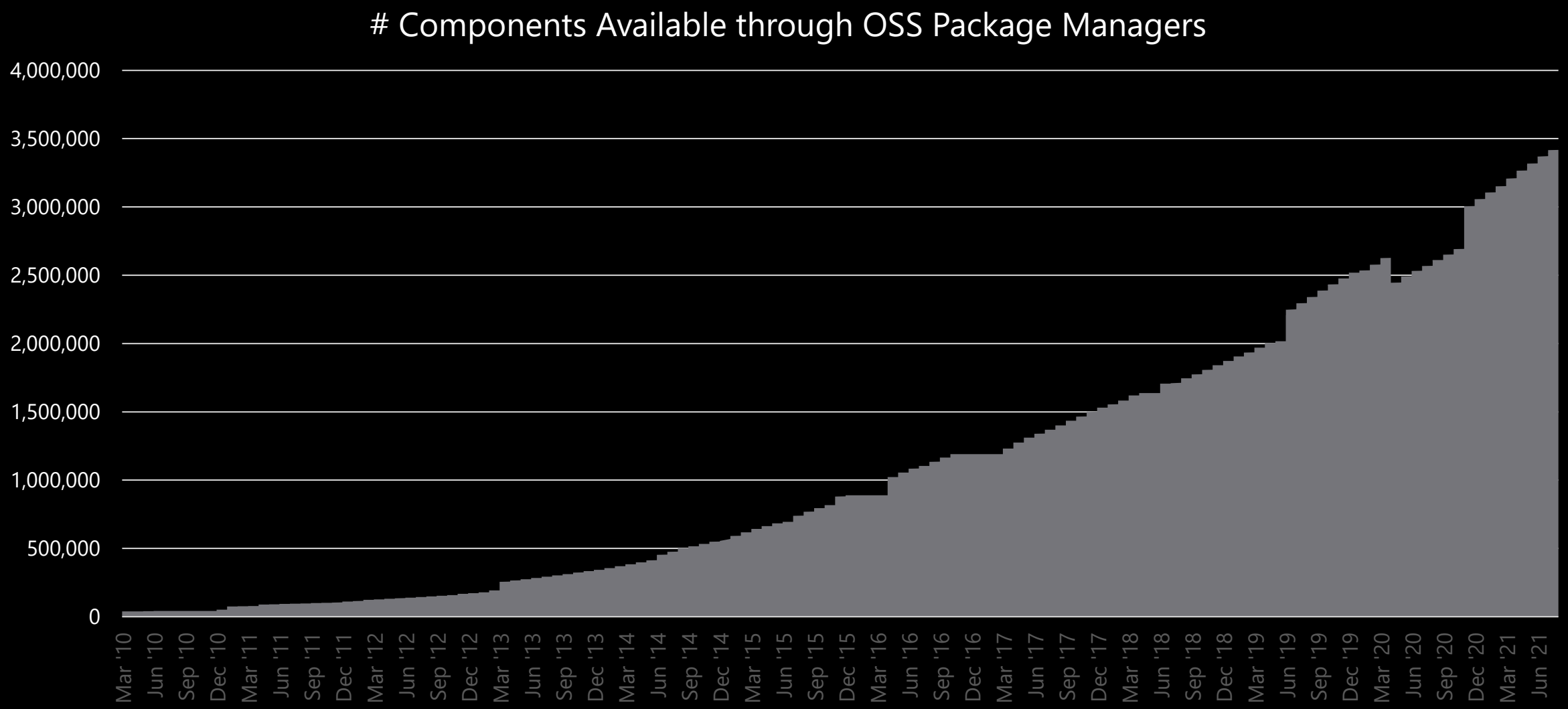
Increasing complexity…

Code Galaxies Visualization (anvaka.github.io)

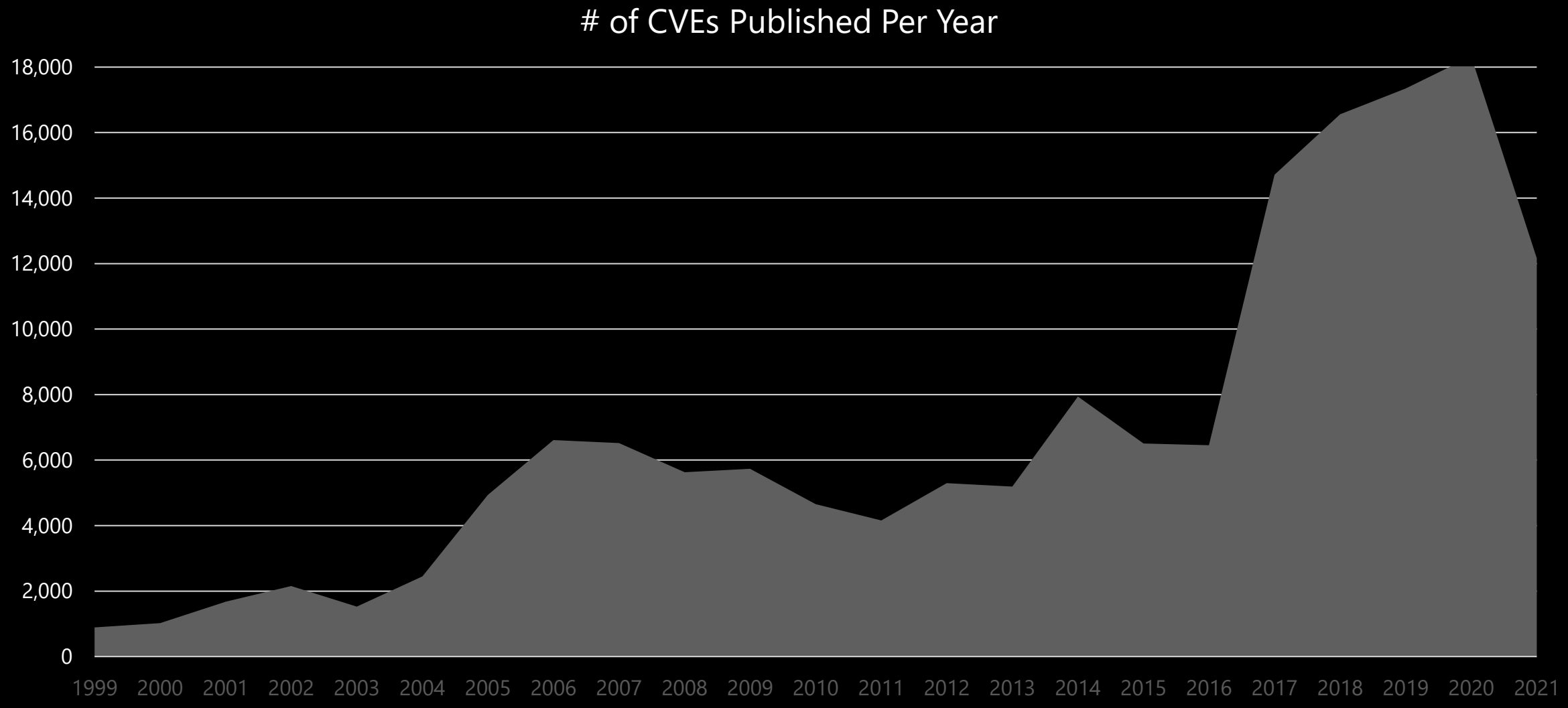


npmgraph - mocha

SCA – Increasing volume of OSS packages



SCA – Increasing volume of vulnerabilities



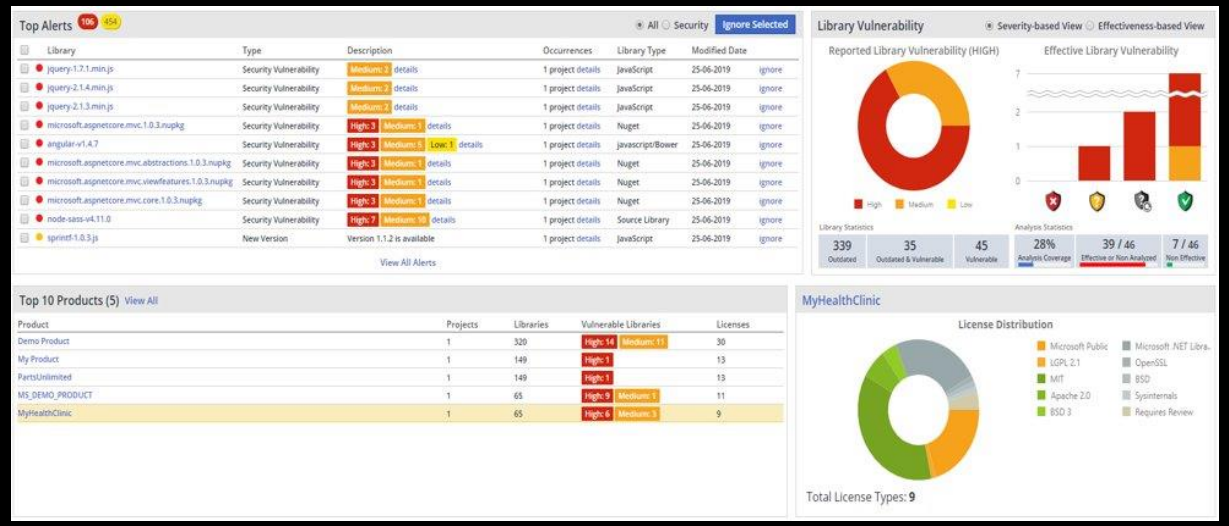
SCA - Governance

Centralize OSS component monitoring
Know which projects are using 3rd party components
Leverage tools and processes to aid in tracking

Understand licensing risks with OSS

- Apache 2.0
- MIT
- BSD 3
- LGPL 2.1
- GPL 3.0
- Microsoft Public
- Public Domain

Ensure the OSS is secure
Respond to Security vulnerabilities



SCA – Why is it so important?

Your company is responsible for everything that ships

An attacker doesn't care who wrote a piece of vulnerable software

Neither will your customers when you must notify them of a breach

Lab – Automate a secure pipeline