HOW TO FIT SEC INTO DEVOPS WITHOUT SECURITY TEAM

Roman Zhukov
Product Security Lead

intel®

DISCLAIMER:

The opinions expressed are solely my own and do not necessarily reflect the official views or opinions of my employer.

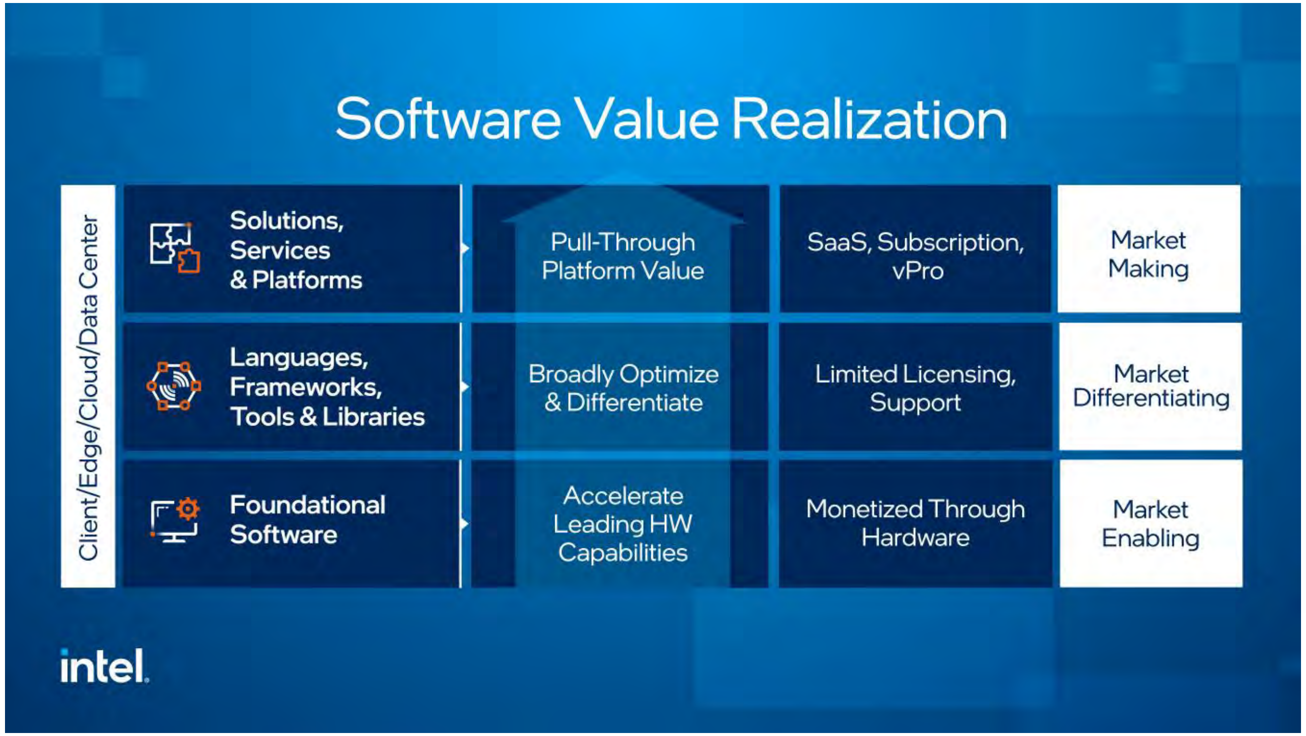
CONF42
DEVSECOPS

ABOUT ME

Roman Zhukov

- 12+ years in Information and Product Security
- Product security expert: SDL, DevSecOps, Architecture
- [former] Brought to market security products and services
- [former] Help to secure large enterprise infrastructures
- Security trainer at Universities
- I am fond of Hiking, Volleyball, Running, Bikes

SOFTWARE @ INTEL

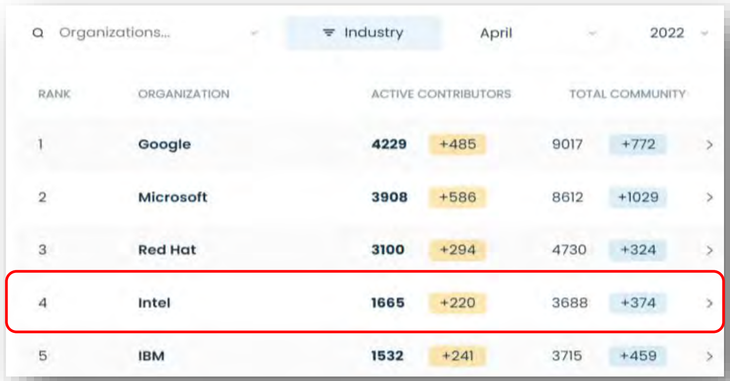


Software Value Realization

Client/Edge/Cloud/Data Center

Solutions, Services & Platforms	Pull-Through Platform Value	SaaS, Subscription, vPro	Market Making
Languages, Frameworks, Tools & Libraries	Broadly Optimize & Differentiate	Limited Licensing, Support	Market Differentiating
Foundational Software	Accelerate Leading HW Capabilities	Monetized Through Hardware	Market Enabling

intel.



Organizations... Industry April 2022

RANK	ORGANIZATION	ACTIVE CONTRIBUTORS		TOTAL COMMUNITY	
1	Google	4229	+485	9017	+772
2	Microsoft	3908	+586	8612	+1029
3	Red Hat	3100	+294	4730	+324
4	Intel	1665	+220	3688	+374
5	IBM	1532	+241	3715	+459

TOP 4 CONTRIBUTOR TO OPEN SOURCE

Open Source Contributor Index

MYTHS BREAKING

Security team and their tools are “aliens” for R&D.

R&D has their own KPI: product ready or security?

Has the yet another vulnerability discovered? Pfff… no one has ever broken us before.

Security doesn’t contain good metrics or clear value. Could we just complete formal scans?

Security is boring and unnoticeable for everybody.

DEVSECOPS BENEFITS

1 Increasing TTM (Time-to-market)

2 Scaling

3 Flexibility

4 Transparency

5 Trust and brand appeal

DEVSECOPS. FLEXIBILITY

CASE

The community unexpectedly discovered a critically vulnerable and extremely popular 3rd party.

WITH DEVSECOPS:

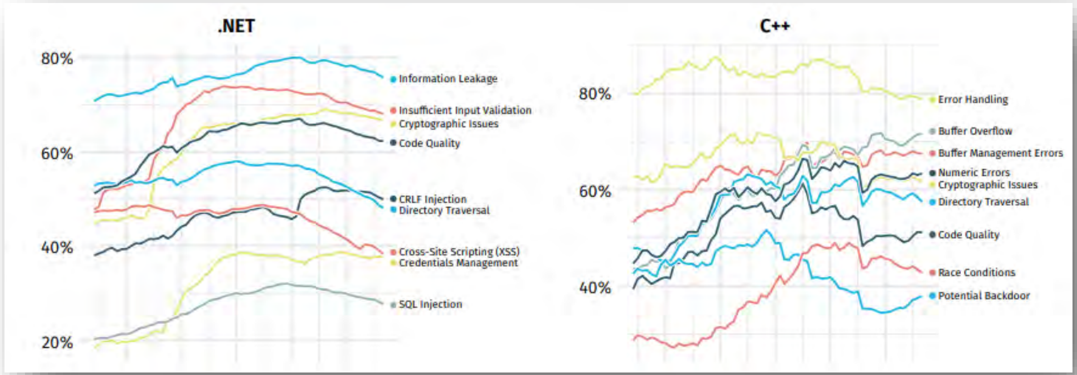
- Thanks to implemented Continuous Security, we understand our components.
- We store logs for previous scans.
- 1 day for infra and product inventory.
- 2-3 days for the out of cycle release, thanks to automation.

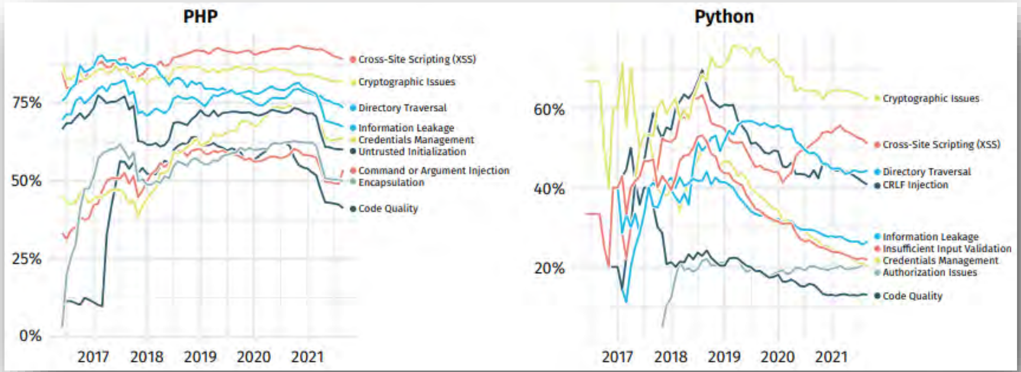
Without DevSecOps:

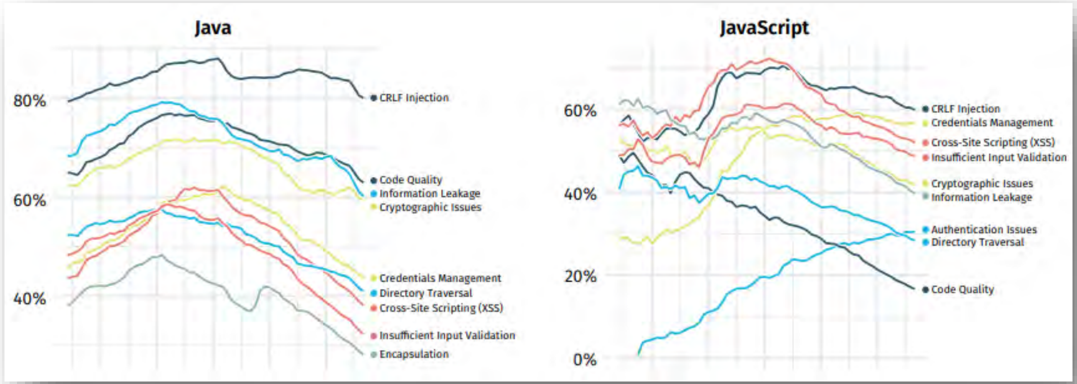
- 3 day for inventory (are we affected?)
- 1 week for the out of cycle release, including approvals, tests and scans.
- additional surprise: living for years 3rd party dependences without updates.

THE MOST POPLAR SECURITY BUGS

State of Software Security, Veracode, February 2022





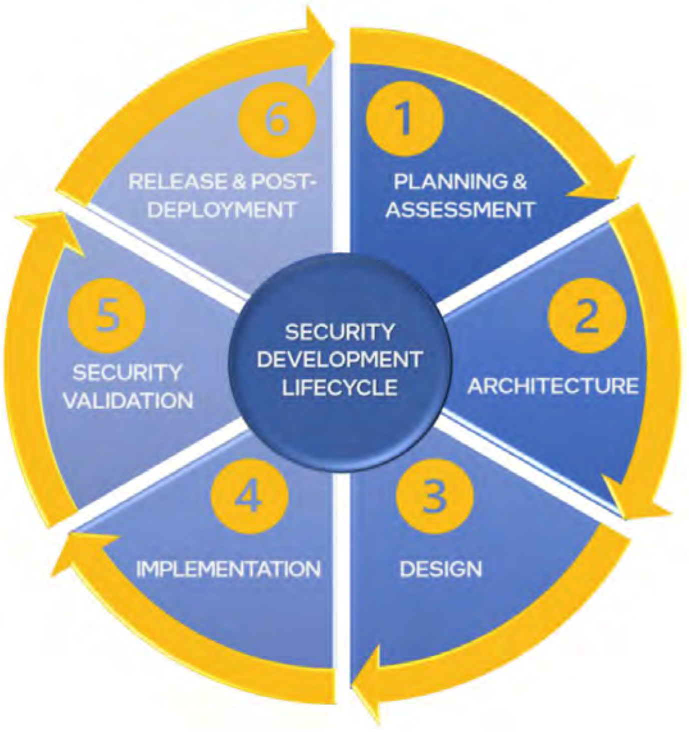


TOP 4:

- ✓ Buffer overflow/underflow
- ✓ Error and input handling
- ✓ Crypto implementation
- ✓ CRLF, XSS and SQLi (web)

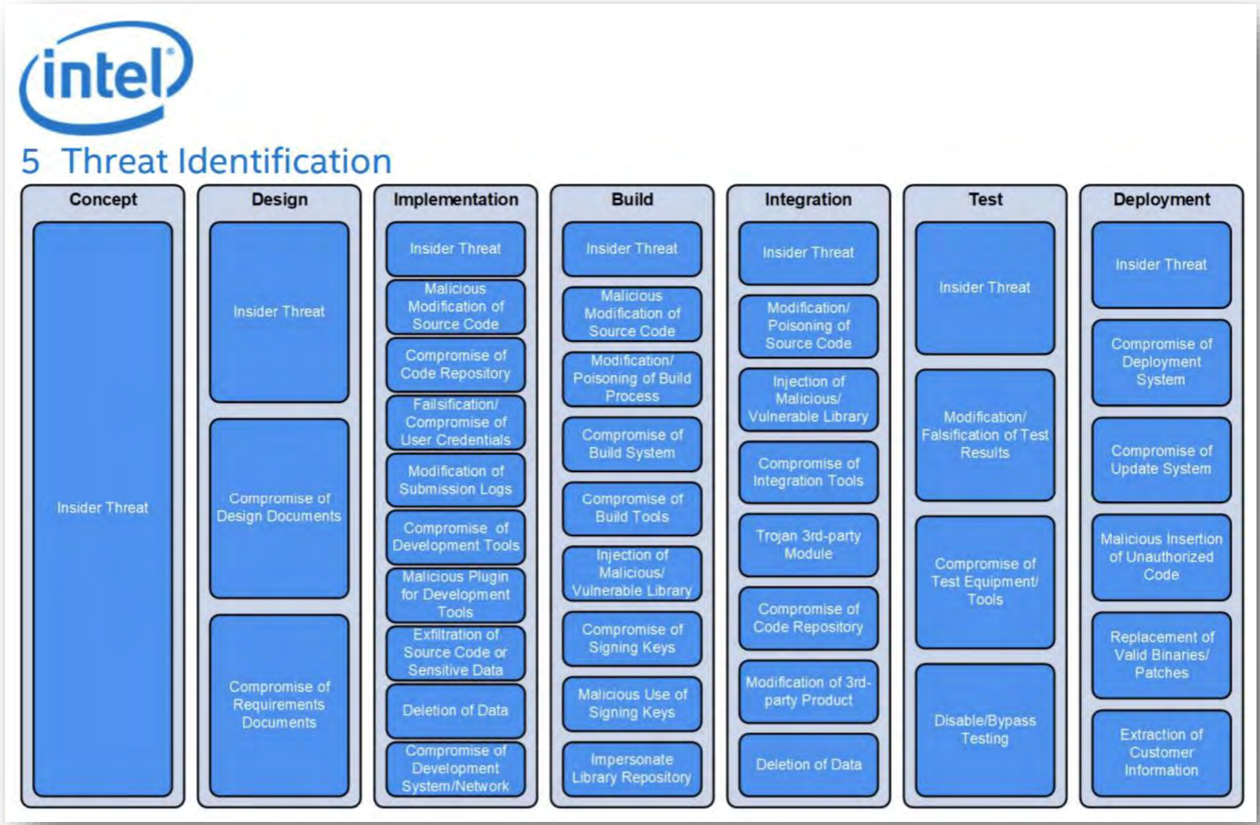
DEVSECOPS AND SW SECURITY @INTEL

Intel® SDL



"Supply Chain Threats – Software" - White Paper, 2021

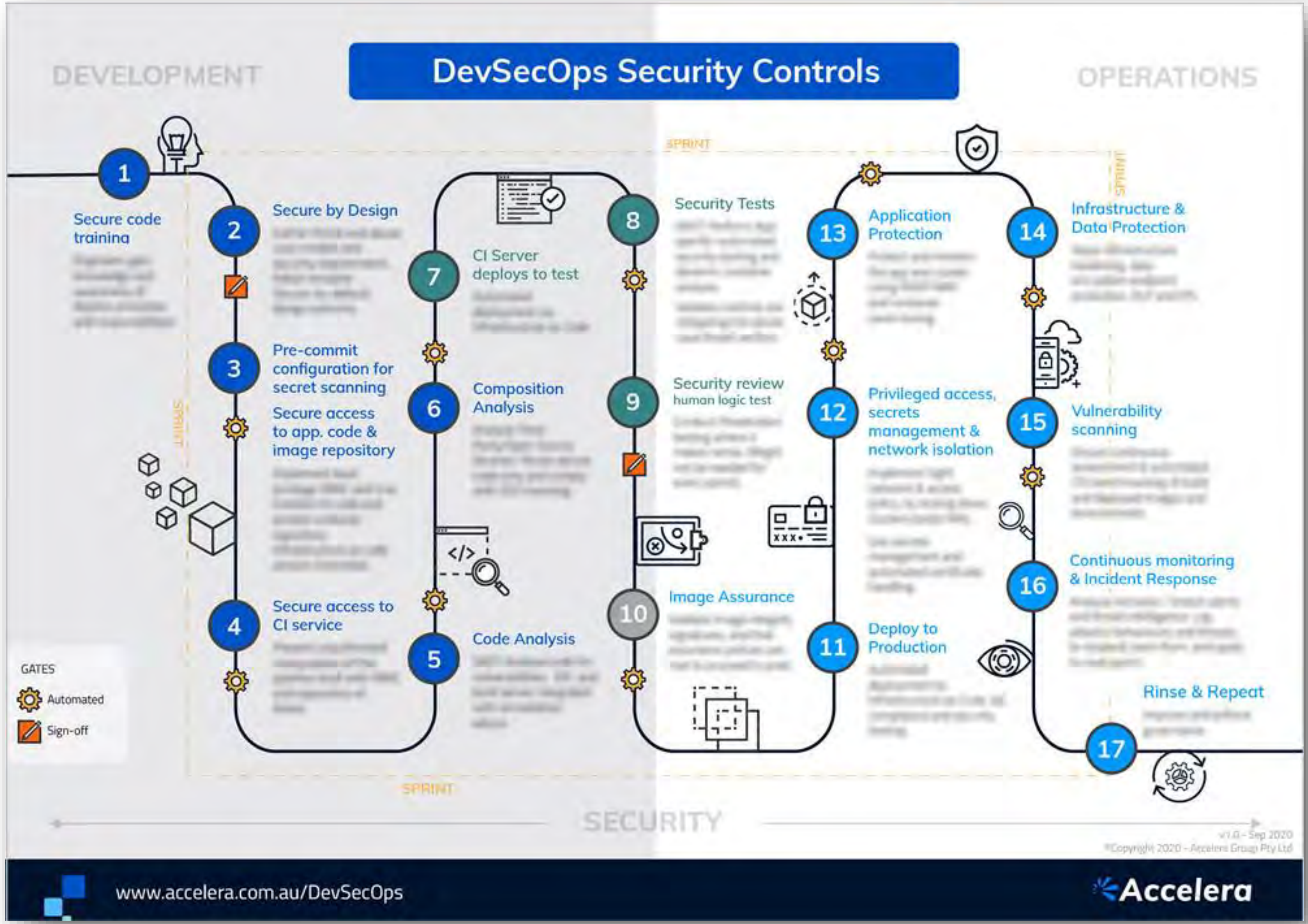
Matthew Areno, PhD, Intel Senior Principal Engineer
Antonio Martin, Intel Principal Engineer



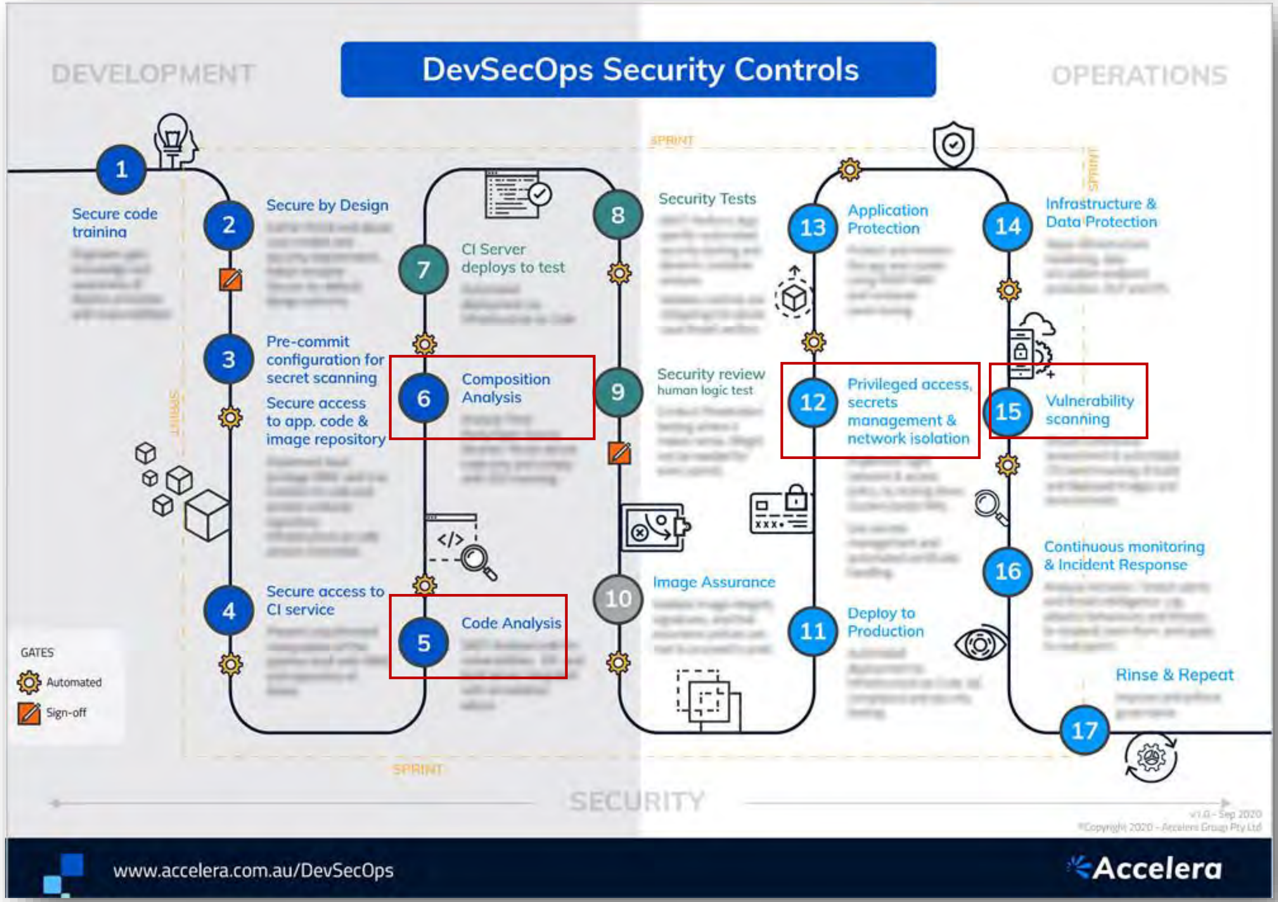
5 Threat Identification

Concept	Design	Implementation	Build	Integration	Test	Deployment
Insider Threat	Insider Threat	Insider Threat	Insider Threat	Insider Threat	Insider Threat	Insider Threat
	Compromise of Design Documents	Malicious Modification of Source Code	Malicious Modification of Source Code	Modification/ Poisoning of Source Code	Modification/ Falsification of Test Results	Compromise of Deployment System
	Compromise of Requirements Documents	Compromise of Code Repository	Modification/ Poisoning of Build Process	Injection of Malicious/ Vulnerable Library	Compromise of Test Equipment/ Tools	Compromise of Update System
		Falsification/ Compromise of User Credentials	Compromise of Build System	Compromise of Integration Tools	Disable/Bypass Testing	Malicious Insertion of Unauthorized Code
		Modification of Submission Logs	Compromise of Build Tools	Trojan 3rd-party Module		Replacement of Valid Binaries/ Patches
		Compromise of Development Tools	Injection of Malicious/ Vulnerable Library	Compromise of Code Repository		Extraction of Customer Information
		Malicious Plugin for Development Tools	Compromise of Signing Keys	Modification of 3rd-party Product		
		Exfiltration of Source Code or Sensitive Data	Malicious Use of Signing Keys	Deletion of Data		
		Deletion of Data	Impersonate Library Repository			
		Compromise of Development System/Network				

AND THEN IT COMES... SECURITY...


DEVELOPMENT
DevSecOps Security Controls
OPERATIONS
SPRINT
1 Secure code training
2 Secure by Design
3 Pre-commit configuration for secret scanning
Secure access to app. code & image repository
4 Secure access to CI service
5 Code Analysis
6 Composition Analysis
7 CI Server deploys to test
8 Security Tests
9 Security review human logic test
10 Image Assurance
11 Deploy to Production
12 Privileged access, secrets management & network isolation
13 Application Protection
14 Infrastructure & Data Protection
15 Vulnerability scanning
16 Continuous monitoring & Incident Response
17 Rinse & Repeat
GATES
Automated
Sign-off
SECURITY
v1.0 - Sep 2020
©Copyright 2020 - Accelera Group Pty Ltd
www.accelera.com.au/DevSecOps
Accelera

AND THEN IT COMES… SECURITY…

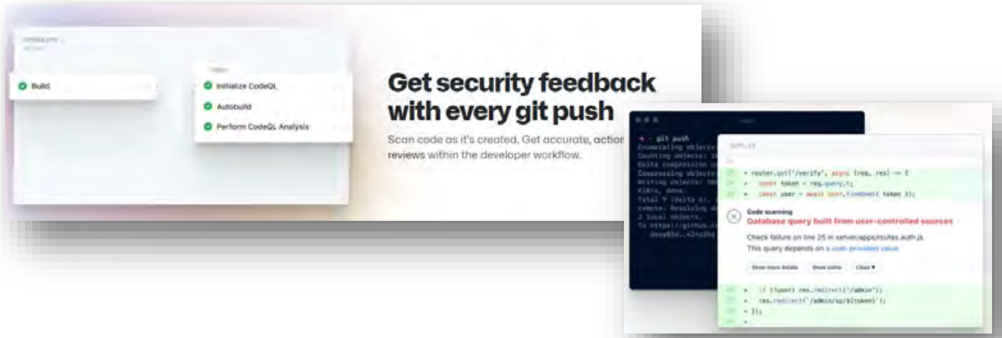

DEVELOPMENT
DevSecOps Security Controls
OPERATIONS
SPRINT
1 Secure code training
2 Secure by Design
3 Pre-commit configuration for secret scanning
Secure access to app. code & image repository
4 Secure access to CI service
5 Code Analysis
6 Composition Analysis
7 CI Server deploys to test
8 Security Tests
9 Security review human logic test
10 Image Assurance
11 Deploy to Production
12 Privileged access, secrets management & network isolation
13 Application Protection
14 Infrastructure & Data Protection
15 Vulnerability scanning
16 Continuous monitoring & Incident Response
17 Rinse & Repeat
GATES
Automated
Sign-off
SECURITY
v1.0 - Sep 2020
©Copyright 2020 - Accelera Group Pty Ltd
www.accelera.com.au/DevSecOps
Accelera

I recommend primarily to focus on:

- SAST (5)
- SCA (6)
- Roles & Secrets (12)
- Vulnerability Scanning (15)
- +2 BONUSES ☺

EXAMPLE: GITHUB

5 - Static Analysis - CodeQL



Build

Initialize CodeQL

Autobuild

Perform CodeQL Analysis

Get security feedback with every git push

Scan code as it's created. Get accurate, action reviews within the developer workflow.

```
~ git push
```

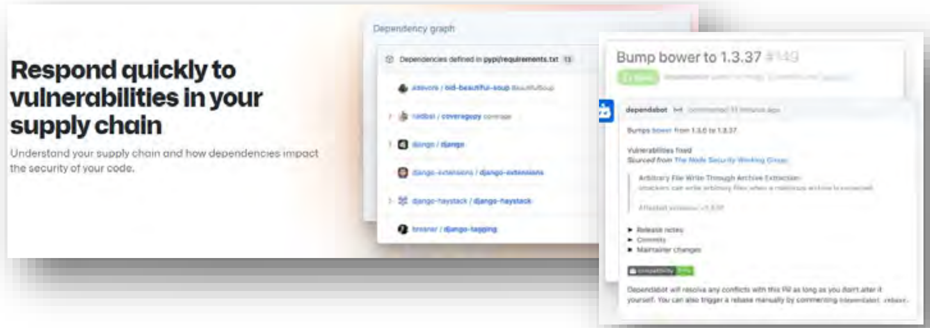
Code scanning

Database query built from user-controlled sources

Check failure on line 25 in server/apps/routes.auth.js

This query depends on a user-provided value.

6 – Software Composition Analysis – DependaBot + RenovateBot



Respond quickly to vulnerabilities in your supply chain

Understand your supply chain and how dependencies impact the security of your code.

Dependency graph

Dependencies defined in pypi/requirements.txt 13

django / django

django-extensions / django-extensions

django-haystack / django-haystack

Bump bower to 1.3.37

dependabot

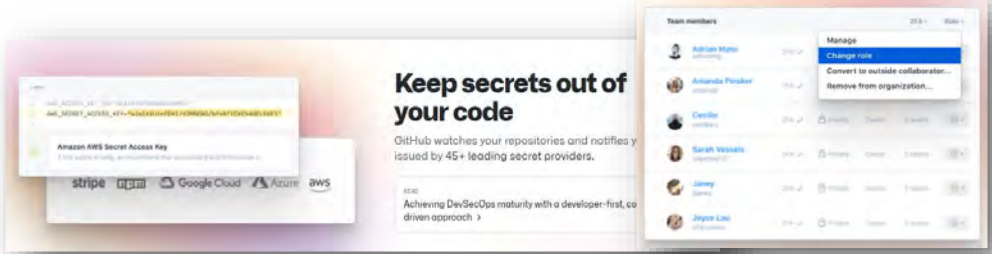
Vulnerabilities fixed

Arbitrary File Write Through Archive Extraction

- Release notes
- Commits
- Maintainer changes

Dependabot will resolve any conflicts with this PR as long as you don't alter it yourself. You can also trigger a rebase manually by commenting @dependabot rebase.

12 – Secret scanning - GitHub Secret scanning, Role management – GitHub roles



Amazon AWS Secret Access Key

Keep secrets out of your code

GitHub watches your repositories and notifies y issued by 45+ leading secret providers.

READ

Achieving DevSecOps maturity with a developer-first, co driven approach ›

Team members

Adrian Mato

Amanda Pinsker

Cecilia

Sarah Vessels

Jamey

Joyce Lau

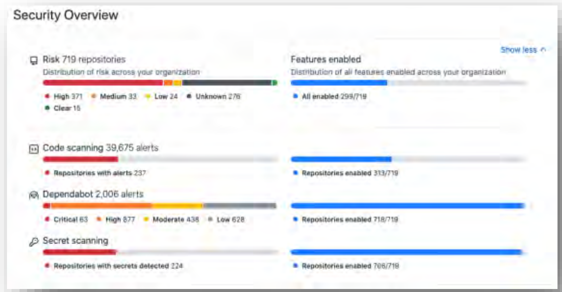
Manage

Change role

Convert to outside collaborator...

Remove from organization...

15 – Vulnerability Scanning: Security overview – for issues in code, External tools - for the end product



Security Overview

Show less

Risk 719 repositories	Features enabled
Distribution of risk across your organization	Distribution of all features enabled across your organization
● High 371 ● Medium 33 ● Low 24 ● Unknown 276 ● Clear 15	● All enabled 299/719
Code scanning 39,675 alerts	
● Repositories with alerts 237	● Repositories enabled 313/719
Dependabot 2,006 alerts	
● Critical 63 ● High 877 ● Moderate 438 ● Low 628	● Repositories enabled 718/719
Secret scanning	
● Repositories with secrets detected 224	● Repositories enabled 706/719

High	0	2	2
High	46	5	14
High		✓	5
Low		0	1

https://github.com/features/security

EXAMPLE: OPEN SOURCE AND FREE TOOLS*

5 - Static Analysis

- SonarQube - free plan exists, dependent on language
- Bandit (not really SAST, but still helps) for Python
- RIPS for PHP
- SemGrep for C#, Go, Java, JavaScript, JSON, Python, Ruby

6 – Software Composition Analysis

- Snyk - free plan exists
- Dependency Track and Dependency Check
- Debricked - free plan exists
- CVE-Bin-tool (by Intel)

12 – Secret scanning

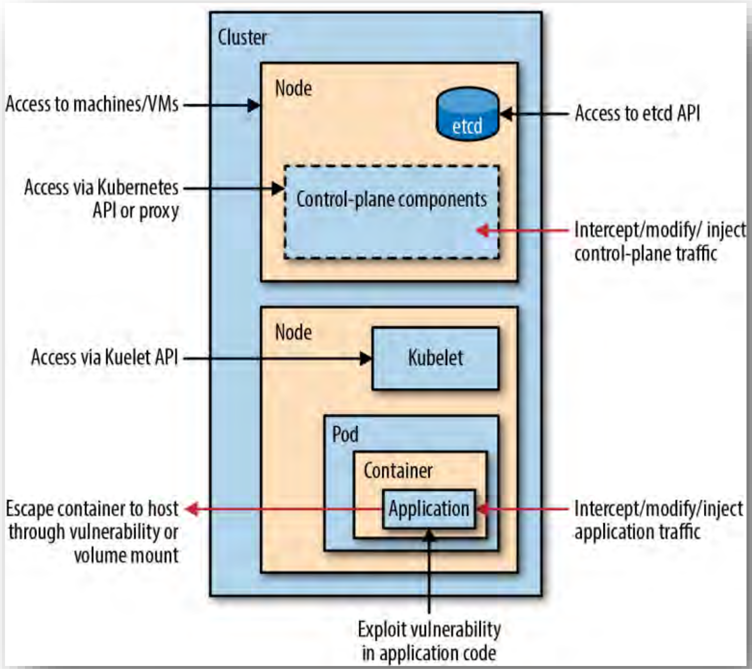
- Gitguardian - free plan exists
- Gitleaks
- Whispers
- Detect-secrets
- Vault – for secret management

15 – Vulnerability Scanning for the end product

- OpenVAS
- Nmap with extentions
- ThreatMapper for outside-in scans
- Nuclei for web servers
- OWASP Zap for web

BONUS #1: CONTAINER AND K8S

What could possibly go wrong…


Cluster
Node
etcd
Control-plane components
Node
Kubelet
Pod
Container
Application
Access to machines/VMs
Access via Kubernetes API or proxy
Access via Kuelet API
Escape container to host through vulnerability or volume mount
Access to etcd API
Intercept/modify/ inject control-plane traffic
Intercept/modify/inject application traffic
Exploit vulnerability in application code

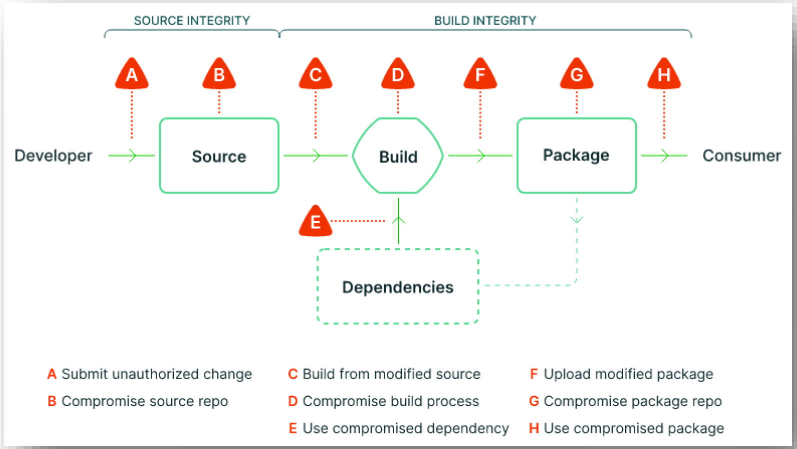
1. Select Images responsibly
   - ✓ Use the image with minimum functionality and packages needed from scratch/distrolles/static/base/busybox (for OS base images).
   - ✓ Verify through Docker Bench, Clair, OpenSCAP and always check Cosign or Docker Content Trust signature.
   - ✓ Apply latest patches and vendor's hardening guides (please READ the Security section at manuals).
   - ✓ Consider to establish the vetting procedure and local repo.
2. Utilize all best practices such as Official Docker security guides and OWASP Cheat Sheet when building containers from Dockerfiles by your own
   - ✓ Run scanning tools such as Hadolint, Dockle, Trivy or KubeLinter.
   - ✓ Configure in rootless mode and avoid privileged Containers.
   - ✓ Improve Container isolation and Restrict all sensitive actions like system calls.
3. Verify secrets are REALLY removed from any type of artifact: YAMLs, Container images, Layers and Helm charts, Environment vars, Public Issues, Release Notes.
4. Consider Runtime security by applying all official Kubernetes Security guides, OpenShift Security guide, Rancher Security guide, NSA/CISA K8s Hardening guide
   - ✓ Implement Service Mesh concept (Istio) and leverage Policy engine (OPA – Open Policy Agent).
   - ✓ Utilize container-native security tools: Calico (network), Falco (anomalies), Checkov (misconfigurations), Monitoring (Prometheus, Kubescape, Kube-bench, Kube-hunter).

BONUS#2: MAKE ALL ARTIFACTS TRUSTED

SLSA ("salsa") is Supply-chain Levels for Software Artifacts

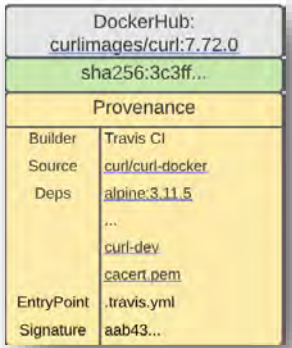
the supply chain is documented, there's infrastructure to generate provenance

the build environment is fully accounted for, dependencies are tracked in provenance and insider threats are ruled out.



SOURCE INTEGRITY
BUILD INTEGRITY
A
B
C
D
F
G
H
Developer
Source
Build
Package
Consumer
E
Dependencies
A Submit unauthorized change
B Compromise source repo
C Build from modified source
D Compromise build process
E Use compromised dependency
F Upload modified package
G Compromise package repo
H Use compromised package

- ✓ Achieve SLSA Level 2 is not a big deal:
  - ✓ Source - Version controlled
  - ✓ Build - Scripted build and Build service
  - ✓ Provenance – Available, Authenticated and Service generated



DockerHub: curlimages/curl:7.72.0	
sha256:3c3ff...	
Provenance	
Builder	Travis CI
Source	curl/curl-docker
Deps	alpine:3.11.5
	...
	curl-dev
	cacert.pem
EntryPoint	.travis.yml
Signature	aab43...

Automate Provenance creation using SLSA GitHub Actions and integrity check with In-Toto attestation tool. Use Cosign for generating and verifying signatures.

THANK YOU!

Reach out to me:

FACEBOOK.COM/R.O.ZHUKOV

ROZHUKOV.BLOGSPOT.COM

LINKEDIN.COM/IN/ROZHUKOV