

DevSecOps: What, Why and How

Anant Shrivastava, NotSoSecure (Claranet Cyber Security)

About: Anant Shrivastava

Director NotSoSecure Global Services

Sysadmin / Development / Security

Project Owner: AndroidTamer, Codevigilant

Contributor : OWASP, null, G4H and more

<https://anantshri.info> (@anantshri on social platforms)

NotSoSecure Global Services (a Claranet group company)

Boutique Consulting firm specialized in training and consulting

Agenda

- **What is DevSecOps?**
- **Why do we need DevSecOps?**
- **How do we do DevSecOps?**
- **Integrate Security in Pipeline**
- **Tools of Trade**
- **Sample Implementation**
- **Case Studies**

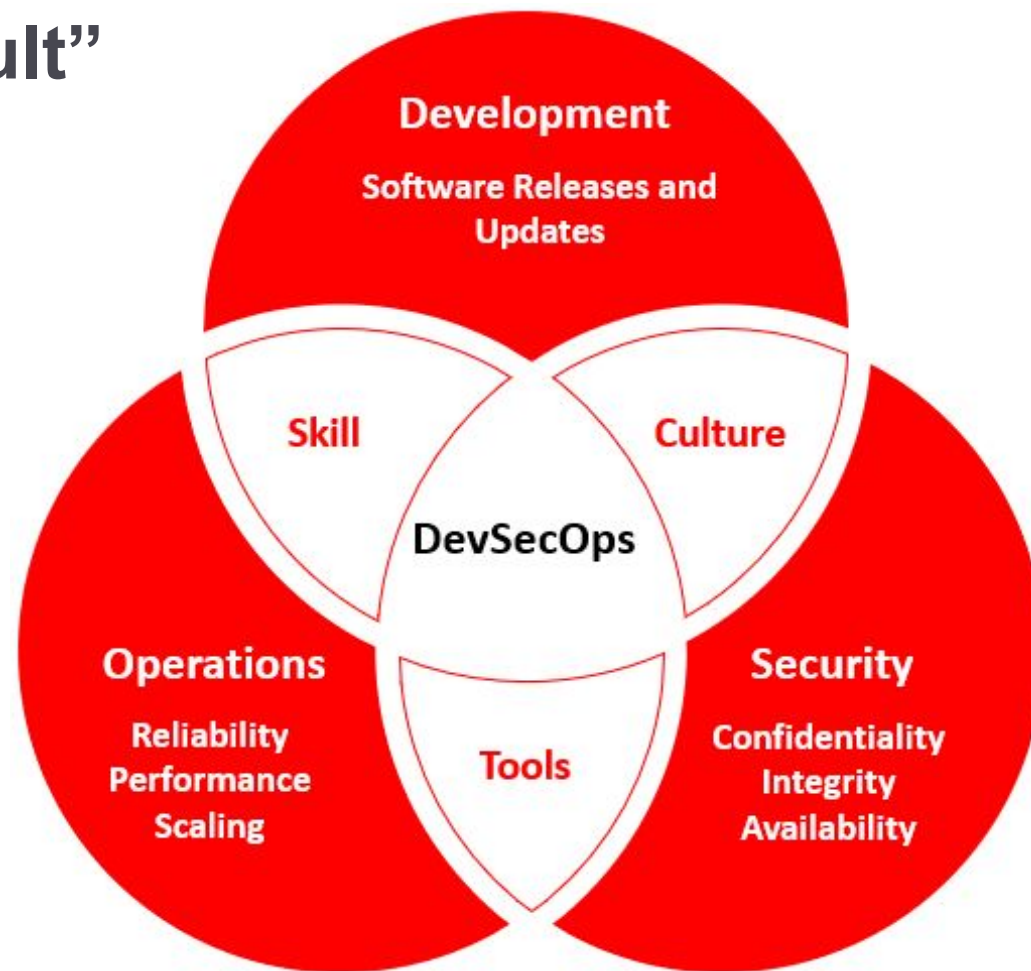
Disclaimer

- I will be listing a lot of tools, It's not an exhaustive list
- I don't endorse or recommend any specific tool / vendor
- Every environment is different: Test and validate before implementing any ideas

What is DevSecOps?

Effort to strive for “Secure by Default”

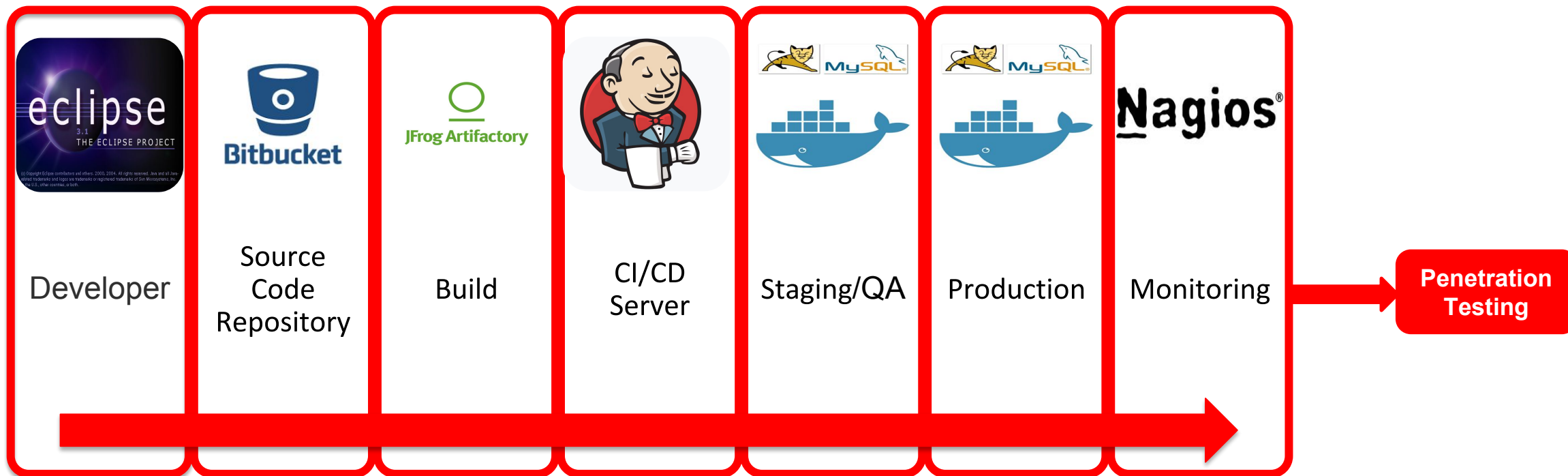
- Integrate Security in **tools**
- Create Security as Code **culture**
- Promote cross **skilling**



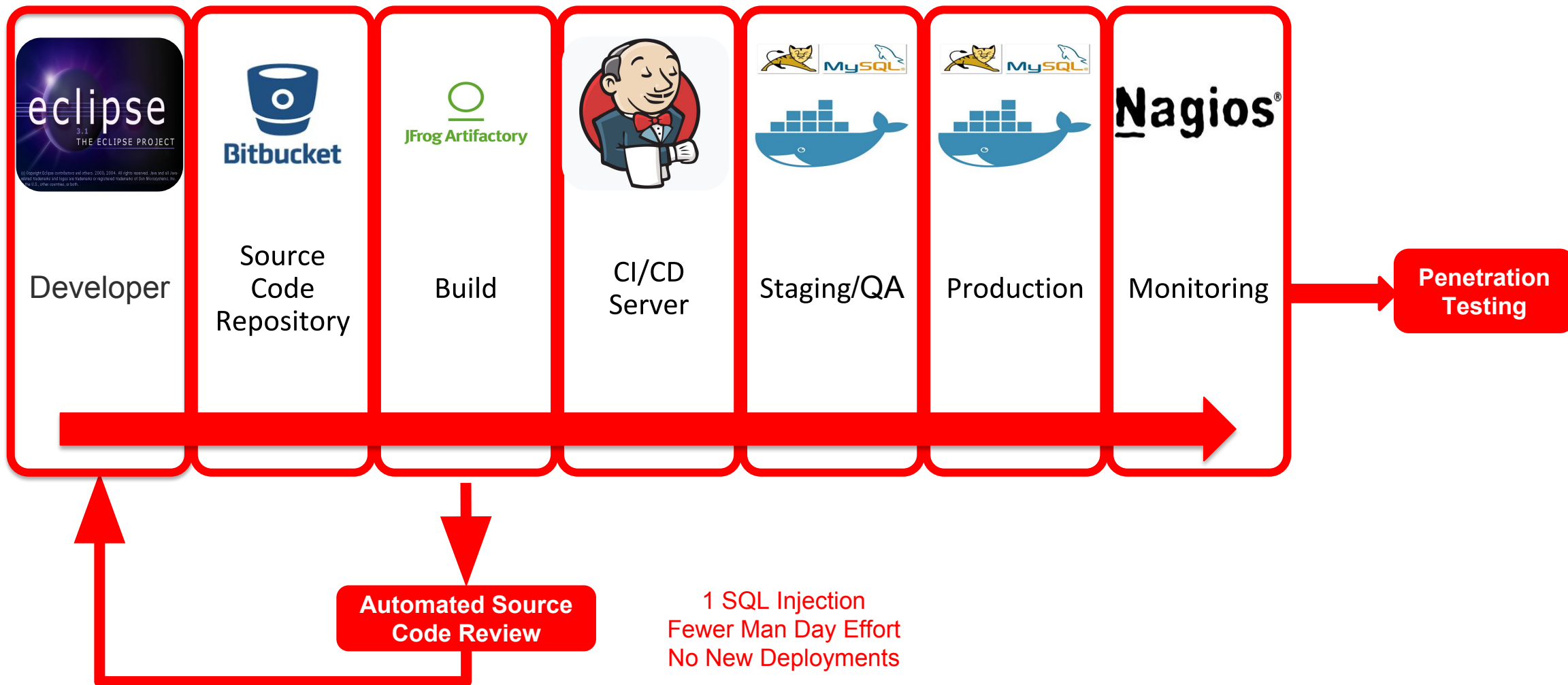
Why do we need DevSecOps?

- DevOps moves at rapid pace, traditional security just can't keep up
- Security as part of process is the only way to ensure safety

Shifting Left saves cost & time



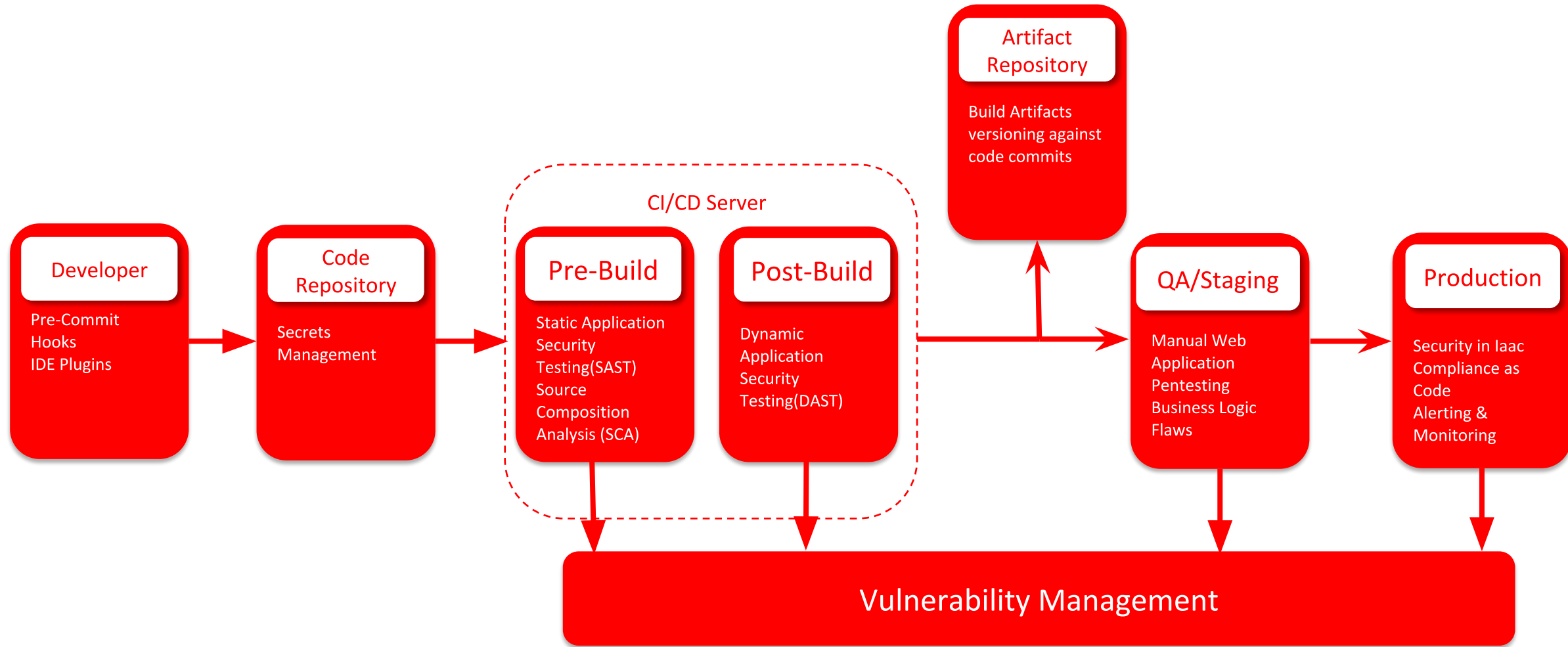
Shifting Left saves cost & time



How do we do DevSecOps?

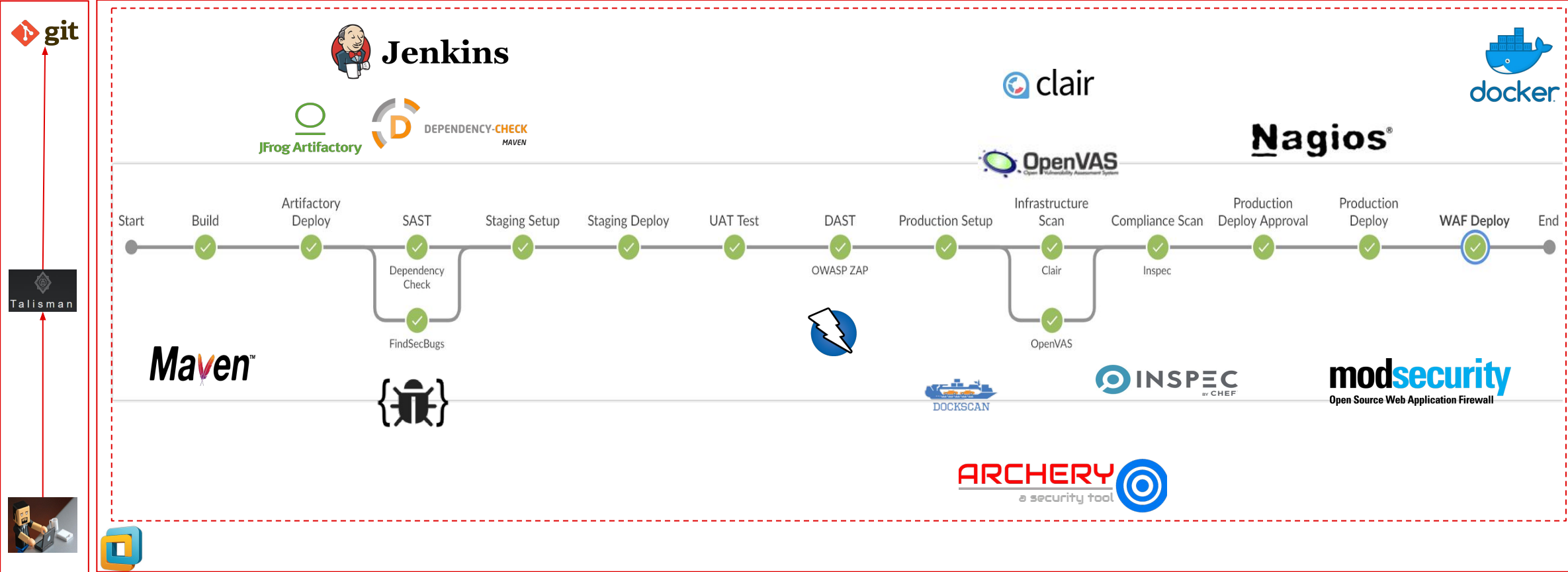
- **DevSecOps is Automation + Cultural Changes**
- **Integrate security into your DevOps Pipeline**
- **Enable cultural changes to embrace DevSecOps**

Injecting Sec in DevOps



Sample Implementation

A simplistic flow of DevSecOps Pipeline using some of the tools mentioned earlier



Tools of the trade

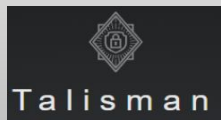
Threat Modelling Tools



ThreatSpec.

Microsoft Threat Modeling Tool

Pre-Commit Hooks



git-secret

truffleHog

Git Hound

Software Composition Analysis



DEPENDENCY-CHECK

Requires.io

Retire.js

Static Analysis Security Testing (SAST)



Bandit



RIPS

sonarqube



IDE Plugins



CAT.net



Secret Management



HashiCorp
Vault

Keywhiz



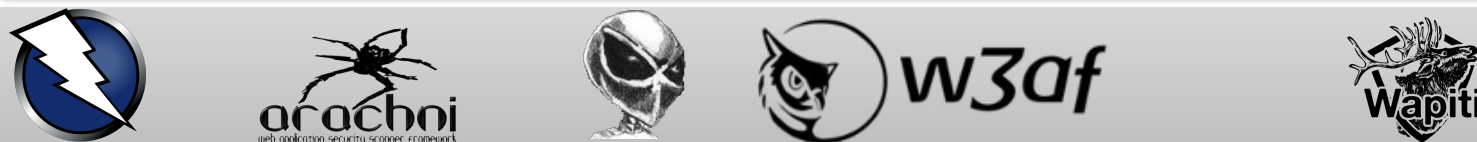
Confidant

Tools of the trade

Vulnerability Management



Dynamic Security Analysis



Infrastructure Scan



Compliance as Code



WAF



To be or not to be in Pipeline

- **API / command line access**
- **Execution start to final output should be 15 minutes max**
- **Containerized / scriptable**
- **Minimal licensing limitations (parallel scans or threads)**
- **Output format parsable / machine readable (no stdout, yes to json /xml)**
- **Configurable to counter false negatives / false positives**

What about Cloud

- **The Threat Landscape changes**
 - Identity and Access Management
 - Billing Attacks
- **Infrastructure as Code allows quick audit / linting**
- **Focus more on:**
 - Security groups
 - Permissions to resources
 - Rouge /shadow admins
 - Forgotten resources (compromises / billing)



Cultural Aspect

- Automation alone will not solve the problems
- Focus on collaboration and inclusive culture
- Encourage security mindset specially if it's outside sec team
- Build allies (security champions) in company
- Avoid Blame Game



Security Champion

- Bridge between Dev, Sec and Ops teams
- Build Security Champions
 - Single Person per team
 - Everyone provided with similar cross skilling opportunities
 - Incentivize other teams to collaborate with Sec team
 - Internal Bug bounties
 - Sponsor Interactions (Parties / get-togethers)
 - Sponsor cross skilling trainings for other teams

Security Enablers

People

- Build relationships between teams, don't isolate
- Identify, nurture security conscious individuals
- Empower Dev/ops to deliver better and faster and secure, instead of blocking.
- Focus on solutions instead of blaming

Process

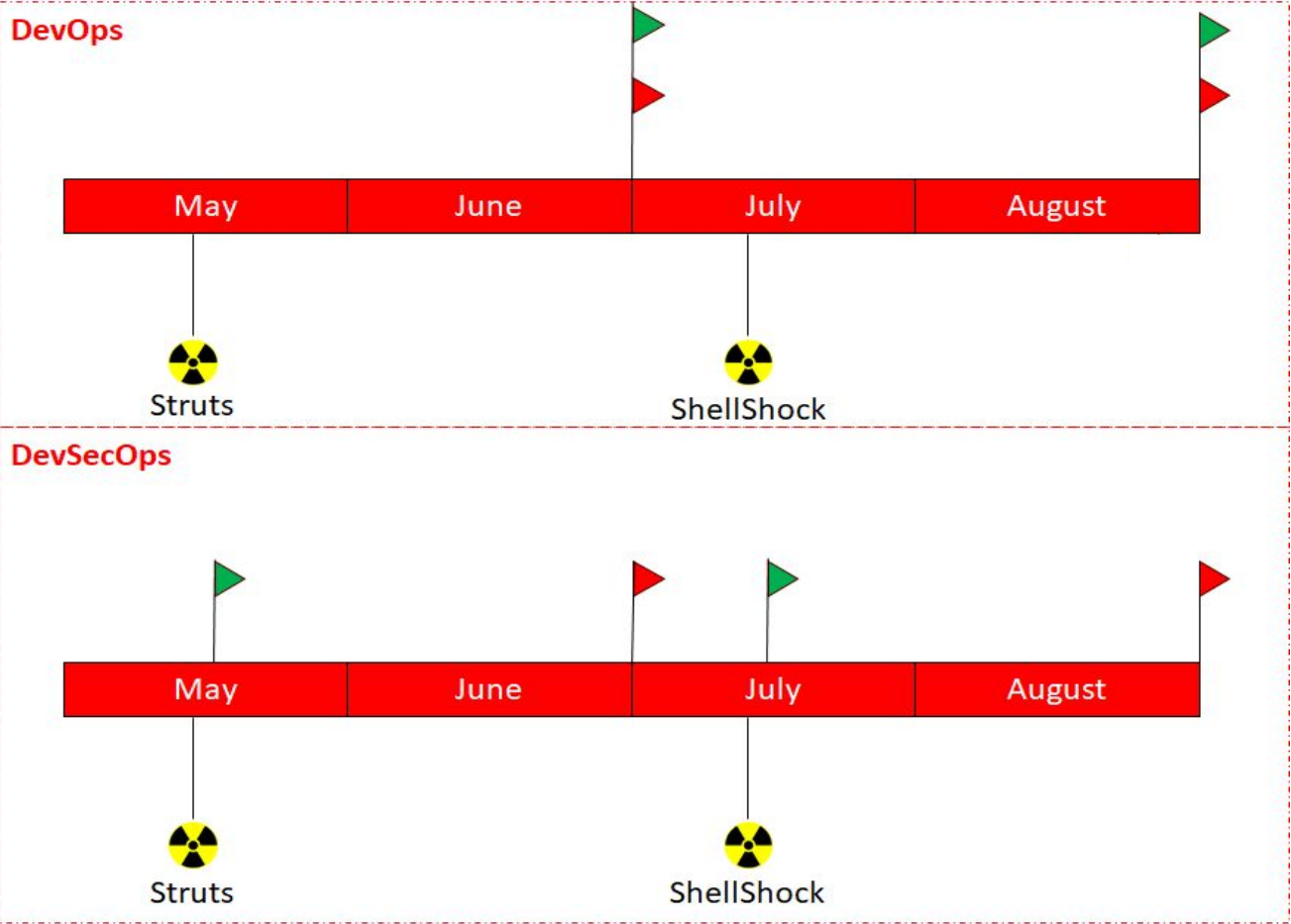
- Involve security from get-go (design or ideation phase)
- Fix by priority, don't attempt to fix it all
- Security Controls must be programmable and automated wherever possible
- DevSecOps Feedback process must be smooth and governed

Technology

- Templatize scripts/tools per language/platform
- Adopt security to devops flow don't expect others to adopt security
- Keep an eye out for simpler and better options and be pragmatic to test and use new tools

Generic Case Study

	Manual Pentest
	Zero Day
	Zero Day Resolved



Case Study

techcrunch.com/2019/01/23/financial-files/



A trove of more than 24 million financial and banking documents, representing tens of thousands of loans and mortgages from some of the biggest banks in the U.S., has been found online after a server security lapse.

The server, running an Elasticsearch database, had more than a decade's worth of data, containing loan and mortgage agreements, repayment schedules and other highly sensitive financial and tax documents that reveal an intimate insight into a person's financial life.

But it wasn't protected with a password, allowing anyone to access and read the massive cache of documents.

It's believed that the database was only exposed for two weeks — but long enough for independent security researcher **Bob Diachenko** to find the data. At first glance, it wasn't immediately known who owned the data. After we inquired with several banks whose customers information was found on the server, the database was shut down on January 15.

Prevention: Recurring Asset Inventory and Automated Assessments

Case Study

#396467 Github Token Leaked publicly for https://github.sc-corp.net Share

State

● Resolved (Closed)

Disclosed

October 8, 2018 6:27pm +0530

Reported To

Snapchat

Asset

app.snapchat.com
(Domain)

Weakness

Cleartext Storage of Sensitive Information




Bounty

\$15,000

Severity

Critical (9.8)

Participants



Visibility


Disclosed (Full)

Prevention:

Pre-commit Hook and continuous repository monitoring

Collapse

More Case Studies

→   bleepingcomputer.com/news/security/7-percent-of-all-amazon-s3-servers-are-exposed-explaining-recent-surge-of-data-leaks/

- ♦ Top defense contractor [Booz Allen Hamilton](#) leaks 60,000 files, including employee security credentials and passwords to a US government system.
- ♦ Verizon partner leaks personal records of [over 14 million Verizon customers](#), including names, addresses, account details, and for some victims — account PINs.
- ♦ An AWS S3 server leaked the personal details of [WWE fans](#) who registered on the company's sites. 3,065,805 users were exposed.
- ♦ Another AWS S3 bucket leaked the personal details of [over 198 million American voters](#). The database contained information from three data mining companies known to be associated with the Republican Party.
- ♦ [Another S3 database](#) left exposed only leaked the personal details of job applications that had Top Secret government clearance.
- ♦ [Dow Jones](#), the parent company of the Wall Street Journal, leaked the personal details of 2.2 million customers.
- ♦ Omaha-based voting machine firm Election Systems & Software (ES&S) left a database exposed online that contained the personal records of [1.8 million Chicago voters](#).
- ♦ Security researchers discovered a Verizon AWS S3 bucket containing over 100 MB of data about the [company's internal system](#) named Distributed Vision Services (DVS), used for billing operations.
- ♦ An [auto-tracking company](#) leaked over a half of a million records with logins/passwords, emails, VIN (vehicle identification number), IMEI numbers of GPS devices and other data that is collected on their devices, customers and auto dealerships.

Prevention: Continuous monitoring and review of cloud assets and config

Case Study: Last one I promise

HackerOne, Inc. [US] | hackerone.com/reports/167859

SUMMARY BY ZOMATO



An alpha version of our Base product was exposed on a Jenkins server.

Thanks @n0rb3r7 for reporting this.

SUMMARY BY CHA5M



During my reconnaissance, I discovered via a self-signed SSL certificate with Zomato listed as the organization name. Upon navigating to the server on port 80, I discovered a default Laravel installation. Curious if there was anything else running on the server, I ran a quick port scan at which time I discovered the alternate HTTP port 8081.

After navigating to port 8081, I discovered that there was a completely open Jenkins instance, which was authenticated to multiple Github accounts and included the complete Zomato base alpha version Android, Dashboard, and Laravel source code. Included in this source were the keys to a Zomato base alpha MySQL server, SMTP server, and SMS service.

The end result of this was a complete database and email takeover of the Zomato base alpha, as well as full access to the Zomato base alpha APK source code. @vinothzomato addressed this issue in a timely manner.

Prevention: Patching and Continuous monitoring of Assets

Is it Enough?

- Rite of passage by periodic pen test and continuous bug bounty
- It's not just important to get feedback but to also action on them
- Risk Acceptance Documentation should be the worst case scenario
not your first bet

The collage displays five screenshots of bug bounty programs and their rules:

- Hackerone - Shopify:** A table listing rewards for various types of vulnerabilities.
- Bugcrowd - Netflix:** A page for the Netflix bug bounty program, managed by Bugcrowd.
- Hackerone - Twitter:** A table listing rewards for various types of vulnerabilities.
- Google - Vulnerability Reward Program (VRP):** A page detailing the rules and scope of the Google VRP.
- Facebook - Responsible Disclosure Policy:** A page detailing Facebook's policy on security vulnerabilities.

Type	Shopify Core*
Arbitrary code execution	\$10,000 - \$25,000
SQL Injection	\$10,000 - \$20,000
Privilege escalation to shop owner	\$5,000 - \$15,000
Authentication bypass - login	\$5,000 - \$10,000
Authentication bypass - app installation	\$2,500 - \$7,500
IDOR / Information Disclosure	\$1,000 - \$5,000
Circumvention of user permission model	\$500 - \$4,000

Category	Examples	Core Twitter[1]	Everything Else
Remote code execution	Command injection	\$20,160	\$10,080
Administrative functionality	Access to internal Twitter applications	\$12,460	\$6,300
Unrestricted access to data (filesystem, database, etc.)	XXE, SQLi	\$12,460	\$6,300
Flaws leaking PII or bypassing significant controls	IDOR, impersonation, sensitive actions by user	\$7,700	\$3,920
Account takeover	OAuth vulnerabilities	\$7,700	\$3,920
Perform activities on	XSS, Android intent		

References

- <https://www.blackhat.com/docs/us-17/thursday/us-17-Lackey-Practical%20Tips-for-Defending-Web-Applications-in-the-Age-of-DevOps.pdf>
- <https://www.sonatype.com/hubfs/2018%20State%20of%20the%20Software%20Supply%20Chain%20Report.pdf>
- <https://snyk.io/opensourcesecurity-2019/>
- <https://www.veracode.com/state-of-software-security-report>

Key Takeaways

- Security is everyone responsibility
- Embrace security as an integral part of the process, use feedback to refine the process
- DevSecOps is not a one size fit all: your mileage will vary