

DevSecOps

What, Why and How

Anant Shrivastava

NotSoSecure Global Services

@anantshri

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

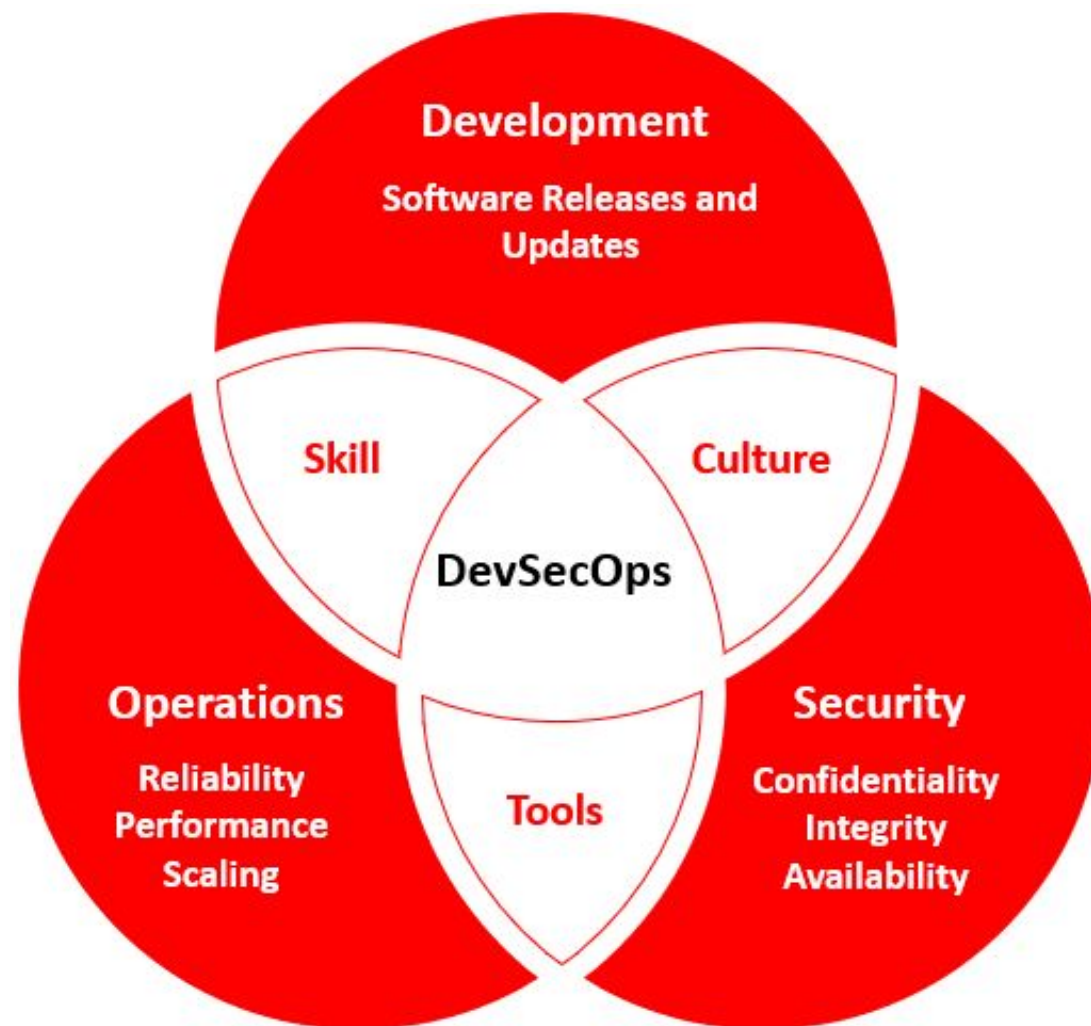
Disclaimer

- 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”

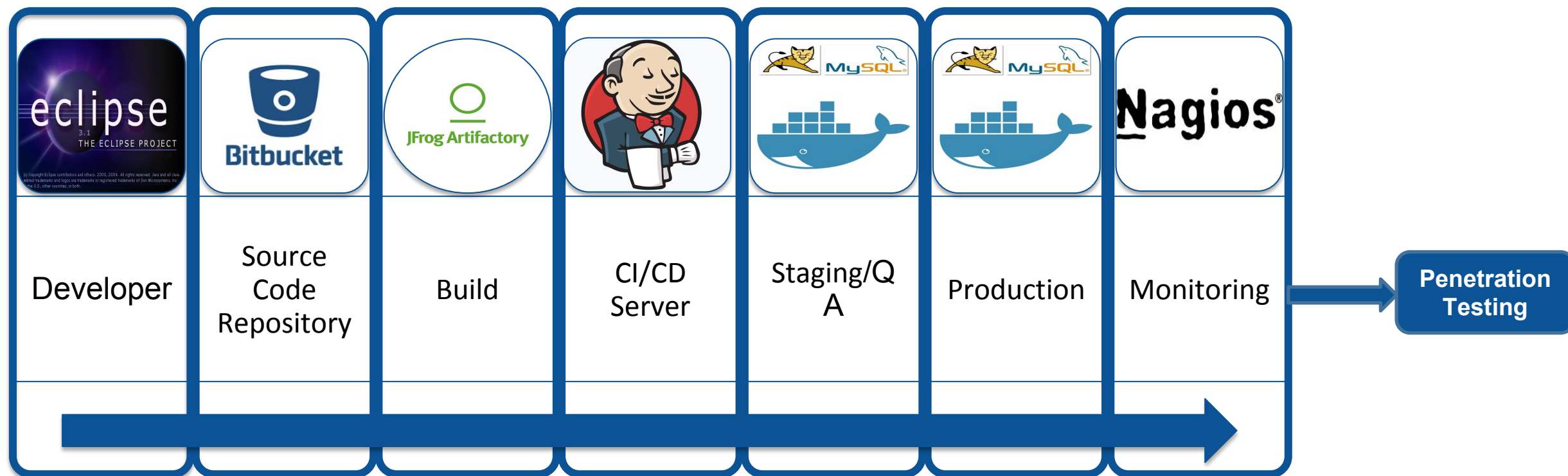
- Integrate Security in **tools**
- Create Security as Code **culture**
- Promote cross **skilling**



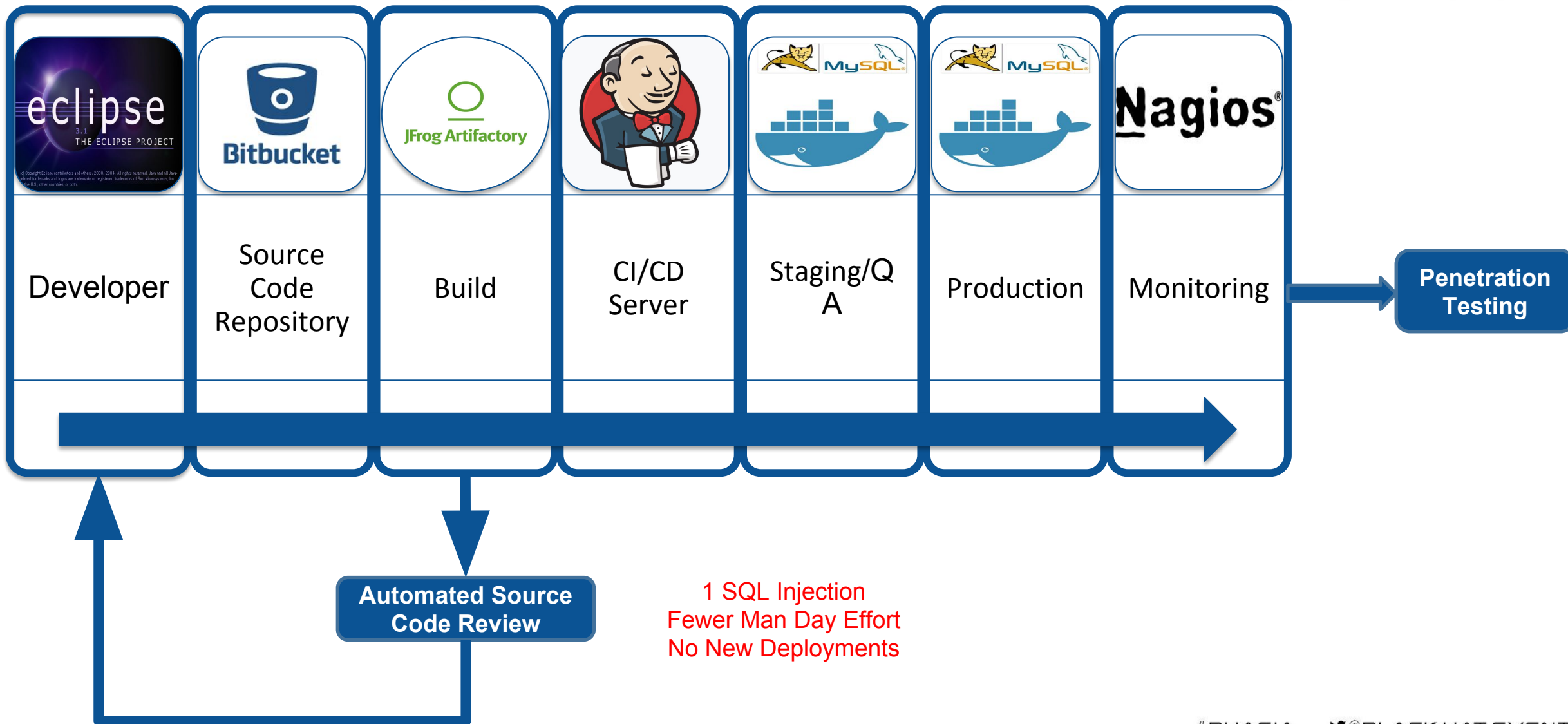
Why do we need DevSecOps

- 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



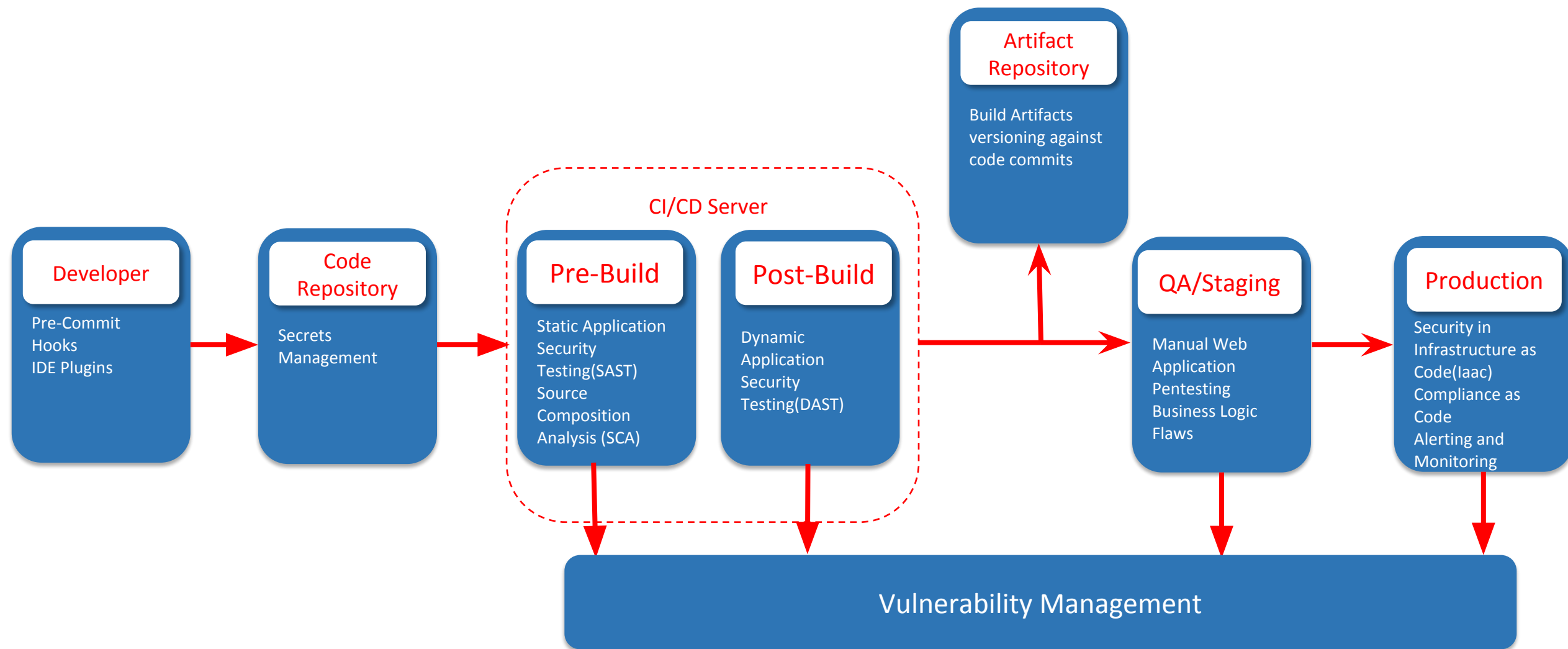
Shifting Left saves cost & time



How do we do DevSecOps

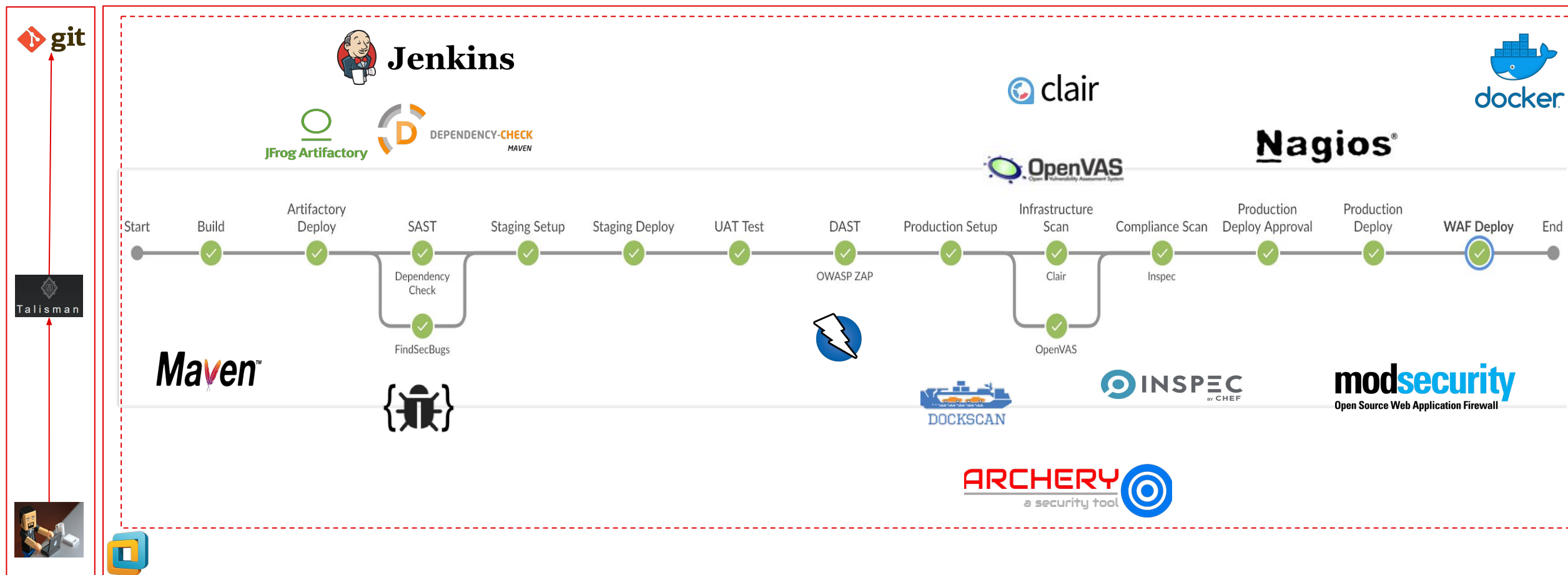
- DevSecOps is Automation + Cultural Changes
- **Integrate security into your DevOps Pipeline**
- Enable cultural changes to embrace DevSecOps

Injecting Sec in DevOps



Sample Implementation

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



Tools of trade

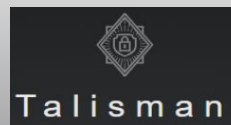
Threat Modelling Tools



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



Keywhiz



Confidant

Tools of trade

Vulnerability Management



Jackhammer



Dynamic Security Analysis



w3af



Infrastructure Scan



OpenVAS
Open Vulnerability Assessment System

anchore



clair



Compliance as Code



DevSec Hardening Framework

Docker Bench for Security

WAF

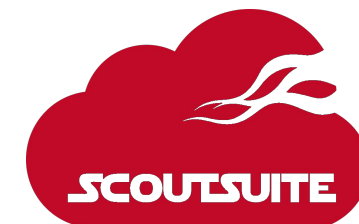


To be or not to be in Pipeline

- 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)






- 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

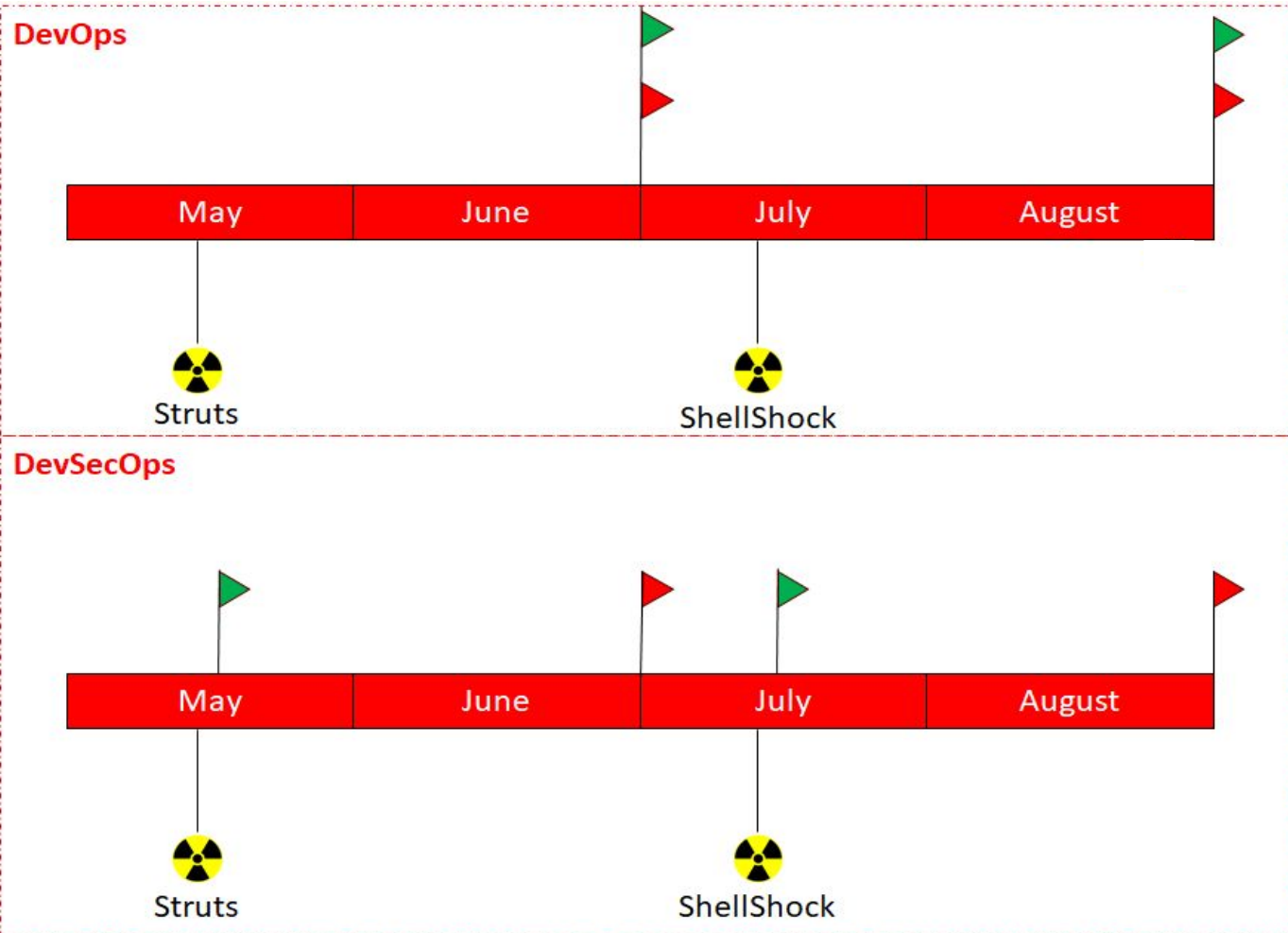
**This is just the tip of the iceberg
(Details out of scope for this session)**



- 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

Generic Case Study

	Manual Pentest
	Zero Day
	Zero Day Resolved



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

#396467

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

Share

State ● Resolved (Closed)

Severity ■■■■■ Critical (9.8)

Disclosed October 8, 2018 6:27pm +0530

Participants 

Reported To [Snapchat](#)

Visibility Disclosed (Full)

Asset app.snapchat.com
(Domain)

Weakness Cleartext Storage of Sensitive Information



Bounty \$15,000

Auth Token accidentally exposed

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 and 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

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



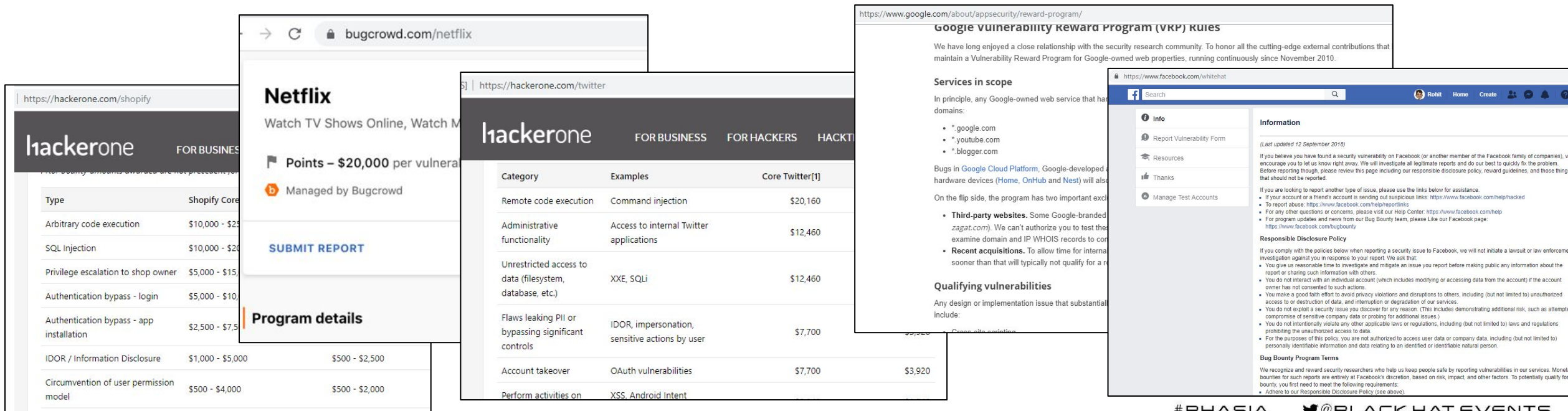
During my reconnaissance, I discovered via a self-signed SSL certificate with Zomato listed as the organization name. Upon navigating to the server on port 80, I discovered a default Laravel installation. Curious if there was anything else running on the server, I ran a quick port scan at which time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

- 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



A collage of screenshots from various bug bounty programs, including Hackerone, Bugcrowd, and Google's VKP, illustrating the diversity of security research opportunities.

Hackerone (https://hackerone.com/shopify)

Type	Shopify Core
Arbitrary code execution	\$10,000 - \$25,000
SQL Injection	\$10,000 - \$25,000
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

Netflix (bugcrowd.com/netflix)

Watch TV Shows Online, Watch Movies

Points – \$20,000 per vulnerability

Managed by Bugcrowd

Program details

hackerone (https://hackerone.com/twitter)

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

Google vulnerability Reward Program (VKP) RULES (https://www.google.com/about/appsecurity/reward-program/)

We have long enjoyed a close relationship with the security research community. To honor all the cutting-edge external contributions that maintain a Vulnerability Reward Program for Google-owned web properties, running continuously since November 2010.

Services in scope

In principle, any Google-owned web service that has public domains:

- * *.google.com
- * *.youtube.com
- * *.blogspot.com

Bugs in **Google Cloud Platform**, Google-developed hardware devices (**Home**, **OnHub** and **Nest**) will also be rewarded.

On the flip side, the program has two important exclusions:

- Third-party websites.** Some Google-branded websites (e.g., **zagat.com**). We can't authorize you to test these sites, examine domain and IP WHOIS records to confirm ownership, or access their databases.
- Recent acquisitions.** To allow time for internal security reviews, we typically do not accept reports for vulnerabilities discovered within 90 days of an acquisition.

Qualifying vulnerabilities

Any design or implementation issue that substantially impacts the security of a service. Examples include:

- Denial of service
- Information disclosure
- Account takeover
- Privilege escalation
- Arbitrary code execution
- SQL injection
- Cross-site scripting
- Cross-site request forgery
- Session hijacking
- Man-in-the-middle
- Man-in-the-browser
- Man-in-the-protocol
- Man-in-the-USB
- Man-in-the-Bluetooth
- Man-in-the-NFC
- Man-in-the-4G/LTE
- Man-in-the-5G
- Man-in-the-6G
- Man-in-the-7G
- Man-in-the-8G
- Man-in-the-9G
- Man-in-the-10G
- Man-in-the-11G
- Man-in-the-12G
- Man-in-the-13G
- Man-in-the-14G
- Man-in-the-15G
- Man-in-the-16G
- Man-in-the-17G
- Man-in-the-18G
- Man-in-the-19G
- Man-in-the-20G
- Man-in-the-21G
- Man-in-the-22G
- Man-in-the-23G
- Man-in-the-24G
- Man-in-the-25G
- Man-in-the-26G
- Man-in-the-27G
- Man-in-the-28G
- Man-in-the-29G
- Man-in-the-30G
- Man-in-the-31G
- Man-in-the-32G
- Man-in-the-33G
- Man-in-the-34G
- Man-in-the-35G
- Man-in-the-36G
- Man-in-the-37G
- Man-in-the-38G
- Man-in-the-39G
- Man-in-the-40G
- Man-in-the-41G
- Man-in-the-42G
- Man-in-the-43G
- Man-in-the-44G
- Man-in-the-45G
- Man-in-the-46G
- Man-in-the-47G
- Man-in-the-48G
- Man-in-the-49G
- Man-in-the-50G
- Man-in-the-51G
- Man-in-the-52G
- Man-in-the-53G
- Man-in-the-54G
- Man-in-the-55G
- Man-in-the-56G
- Man-in-the-57G
- Man-in-the-58G
- Man-in-the-59G
- Man-in-the-60G
- Man-in-the-61G
- Man-in-the-62G
- Man-in-the-63G
- Man-in-the-64G
- Man-in-the-65G
- Man-in-the-66G
- Man-in-the-67G
- Man-in-the-68G
- Man-in-the-69G
- Man-in-the-70G
- Man-in-the-71G
- Man-in-the-72G
- Man-in-the-73G
- Man-in-the-74G
- Man-in-the-75G
- Man-in-the-76G
- Man-in-the-77G
- Man-in-the-78G
- Man-in-the-79G
- Man-in-the-80G
- Man-in-the-81G
- Man-in-the-82G
- Man-in-the-83G
- Man-in-the-84G
- Man-in-the-85G
- Man-in-the-86G
- Man-in-the-87G
- Man-in-the-88G
- Man-in-the-89G
- Man-in-the-90G
- Man-in-the-91G
- Man-in-the-92G
- Man-in-the-93G
- Man-in-the-94G
- Man-in-the-95G
- Man-in-the-96G
- Man-in-the-97G
- Man-in-the-98G
- Man-in-the-99G
- Man-in-the-100G

Facebook (https://www.facebook.com/whitehat)

Search

Info

- Report Vulnerability Form
- Resources
- Thanks
- Manage Test Accounts

Information

(Last updated 12 September 2018)

If you believe you have found a security vulnerability on Facebook (or another member of the Facebook family of companies), we encourage you to let us know right away. We will investigate all legitimate reports and do our best to quickly fix the problem. Before reporting though, please review this page including our responsible disclosure policy, reward guidelines, and those things that should not be reported.

If you are looking to report another type of issue, please use the links below for assistance:

- If your account or a friend's account is sending out suspicious links: <https://www.facebook.com/help/hacked>
- To report abuse: <https://www.facebook.com/help/reportlinks>
- For any other questions or concerns, please visit our Help Center: <https://www.facebook.com/help>
- For program updates and news from our Bug Bounty team, please Like our Facebook page: <https://www.facebook.com/bugbounty>

Responsible Disclosure Policy

If you comply with the policies below when reporting a security issue to Facebook, we will not initiate a lawsuit or law enforcement investigation against you in response to your report. We ask that:

- You give us reasonable time to investigate and mitigate an issue you report before making public any information about the report or sharing such information with others.
- You do not interact with an individual account (which includes modifying or accessing data from the account) if the account owner has not consented to such actions.
- You make a good faith effort to avoid privacy violations and disruptions to others, including (but not limited to) unauthorized access to or destruction of data, and interruption or degradation of our services.
- You do not exploit a security issue you discover for any reason. (This includes demonstrating additional risk, such as attempts to compromise sensitive company data or probing for additional issues.)
- You do not intentionally violate any other applicable laws or regulations, including (but not limited to) laws and regulations prohibiting the unauthorized access to data.
- For the purposes of this policy, you are not authorized to access user data or company data, including (but not limited to) personally identifiable information and data relating to an identified or identifiable natural person.

Bug Bounty Program Terms

We recognize and reward security researchers who help us keep people safe by reporting vulnerabilities in our services. Monetary bounties for such reports are entirely at Facebook's discretion, based on risk, impact, and other factors. To potentially qualify for bounty, you first need to meet the following requirements:

- Adhere to our Responsible Disclosure Policy (see above)

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