RSAConference2018
San Francisco | April 16 – 20 | Moscone Center

NOW MATTERS

SESSION ID: DEV-F01

DOS AND DON'TS OF DEVSECOPS

Hasan Yasar

Technical Manager, Adjunct Faculty Member
CERT | Software Engineering Institute | Carnegie Mellon University
@SecureLifeCycle

Notices

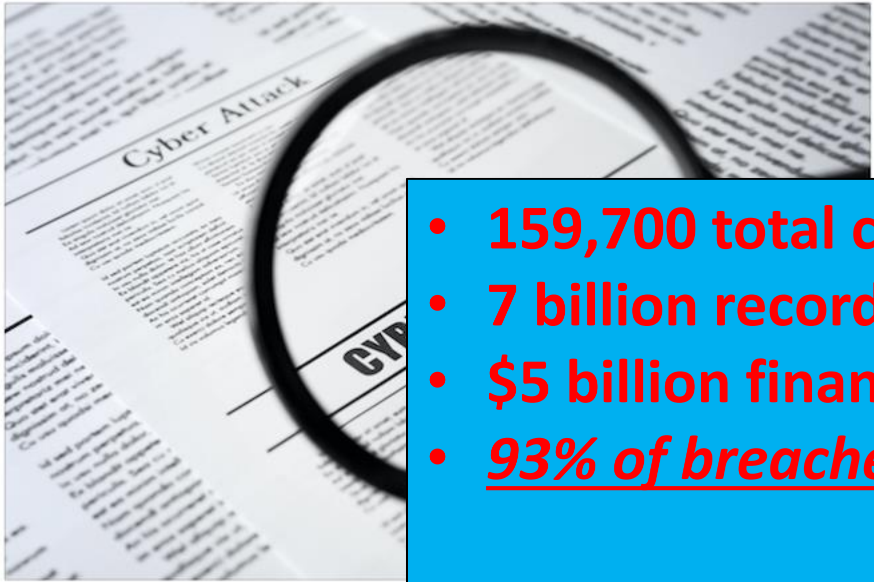
This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

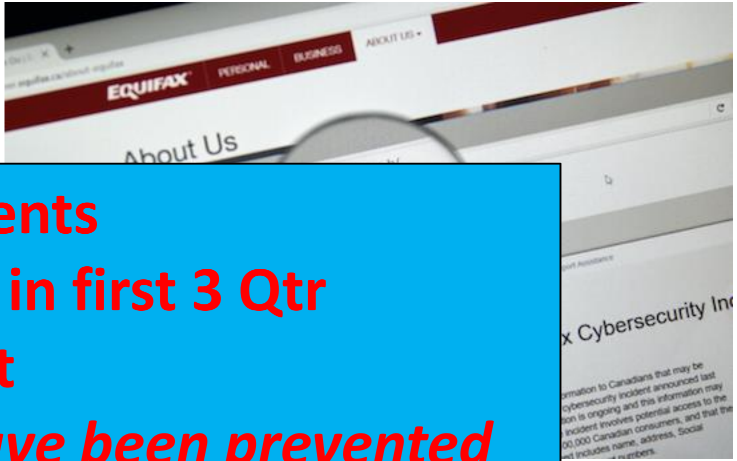
NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

DM18-0424

THE FACT!

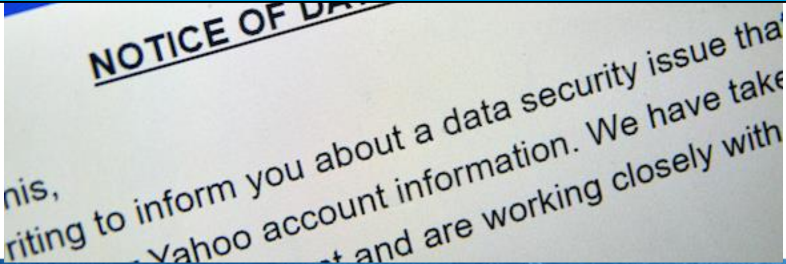
2017 Incident Highlights



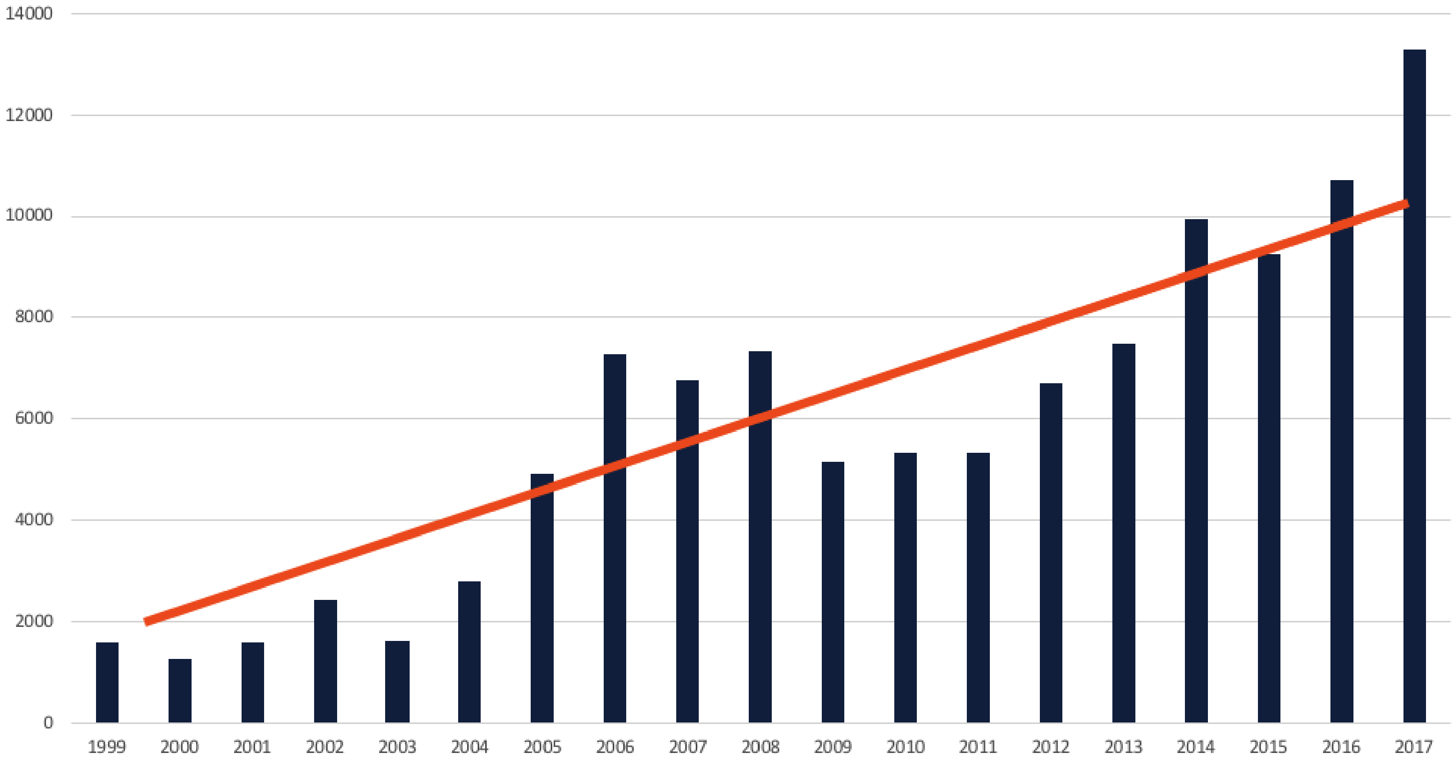


- **159,700 total cyber incidents**
- **7 billion records exposed in first 3 Qtr**
- **$5 billion financial impact**
- ***<u>93% of breaches could have been prevented</u>***

*Online Trust Alliance report 2018


NOTICE OF DA
nis,
riting to inform you about a data security issue tha
Yahoo account information. We have take
and are working closely with

Software Vulnerabilities (CVEs) by Year



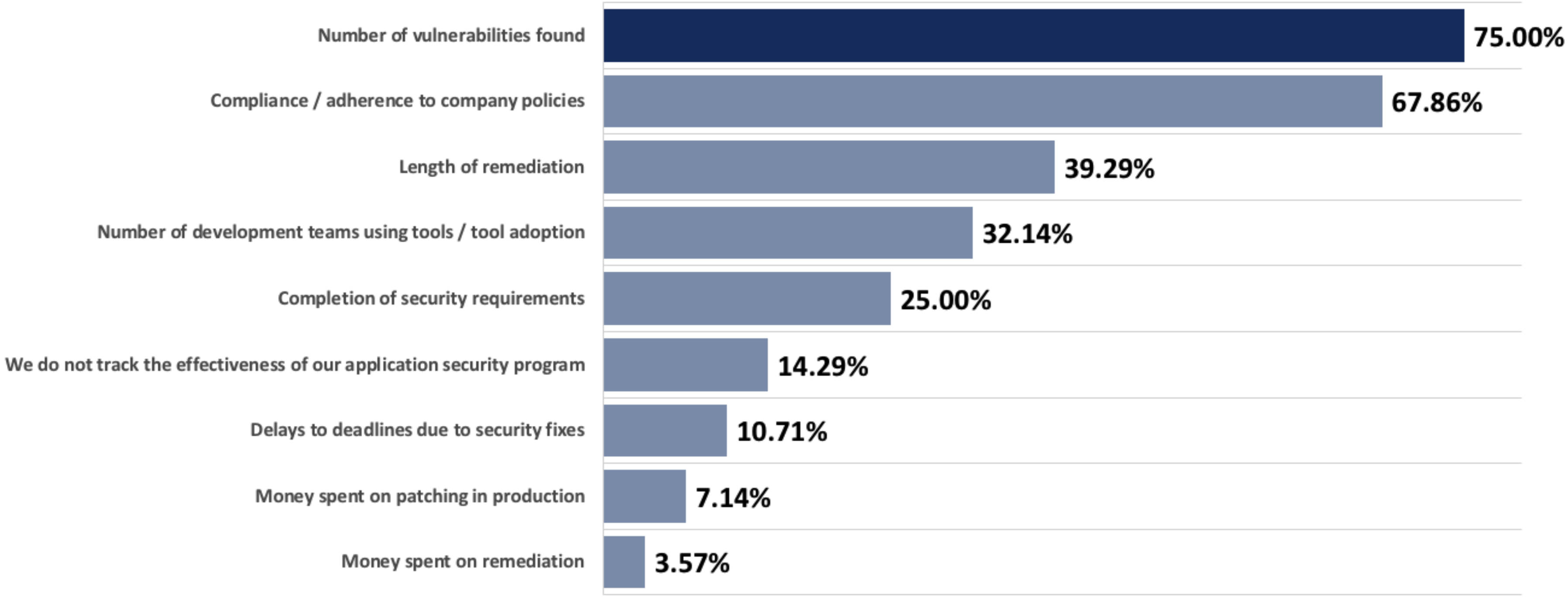
Source: cve.mitre.org as of August 2017

The world we live in..

Software is eating up the world !

Marc Andreessen
https://www.wsj.com/articles/SB10001424053111903480904576512250915629460

How We Manage Software Security - *Application Security Metrics , Financial Institutes*

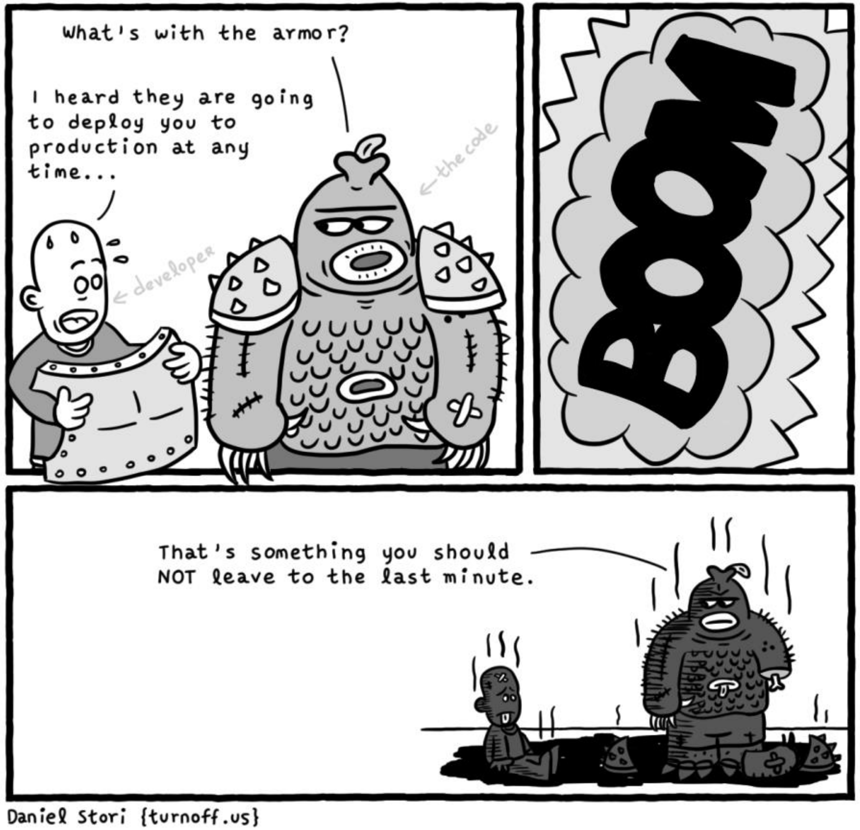


Source: “Managing Application Security”, Security Compass, 2017.

Challenges with Secure Software Development

- Writing code is hard
- Lack of security skills
- Legacy software
- Best practices are insufficient
- Lack of risk focus, lack of audit and control points
- Wrong automated tools
- Unsupervised collaboration
- Emphasis on speed
- Vulnerabilities in deployment pipeline
- Unprotected production environment
- Lack of security requirements traceability

So we all do "Last Minute Security"…


What's with the armor?
I heard they are going to deploy you to production at any time...
← the code
← developer
BOOM
That's something you should NOT leave to the last minute.
Daniel Stori {turnoff.us}

https://dzone.com/articles/last-minute-security-comic

DEVOPS WITH PRINCIPLES

Three Pillars , p^3

DevOps enables “Continuous Everything” on

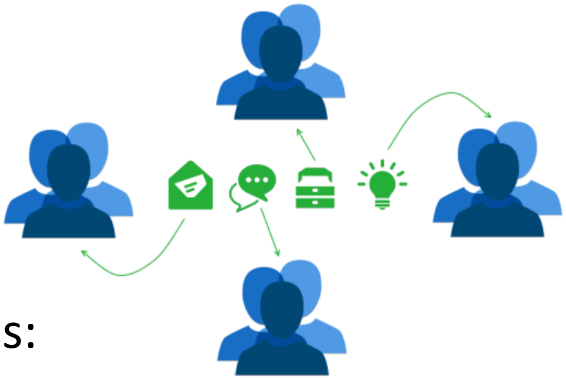
People

Process

Platform

BLUF(Bottom Line Up Front) : People

- Heavy collaboration between all stakeholders
  - Secure Design / Architecture decisions
  - Secure Environment / Network configuration
  - Secure Deployment planning
  - Secure Code Review
- Constantly available open communication channels:
  - Dev and OpSec together in all project decision meeting
  - Chat/e-mail/Wiki services available to all team members

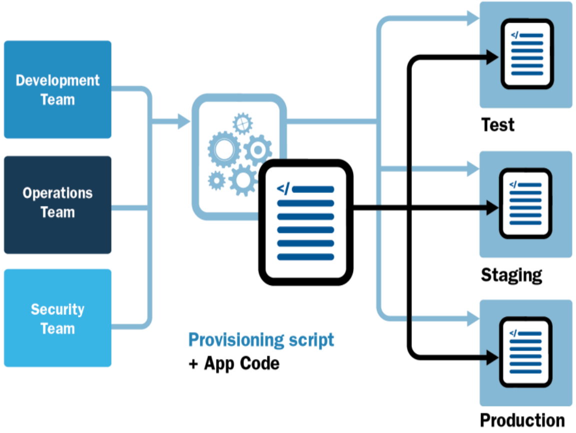


BLUF: Process

- Establish a *process* to enable *people* to succeed using the *platform* to develop secure application
- Such that;
  - Constant communication and visible to all
  - Ensures that tasks are testable and repeatable
  - Frees up human experts to do challenging, creative work
  - Allows tasks to be performed with minimal effort or cost
  - Creates confidence in task success, after past repetitions
  - Faster deployment , frequent quality release

BLUF: Platform

- Where *people* use *process* to build secure software
  - Automated environment creation and provisioning
  - Automated infrastructure testing
  - Parity between Development, QA, Staging, and Production environments
  - Sharing and versioning of environmental configurations
  - Collaborative environment between all stakeholders

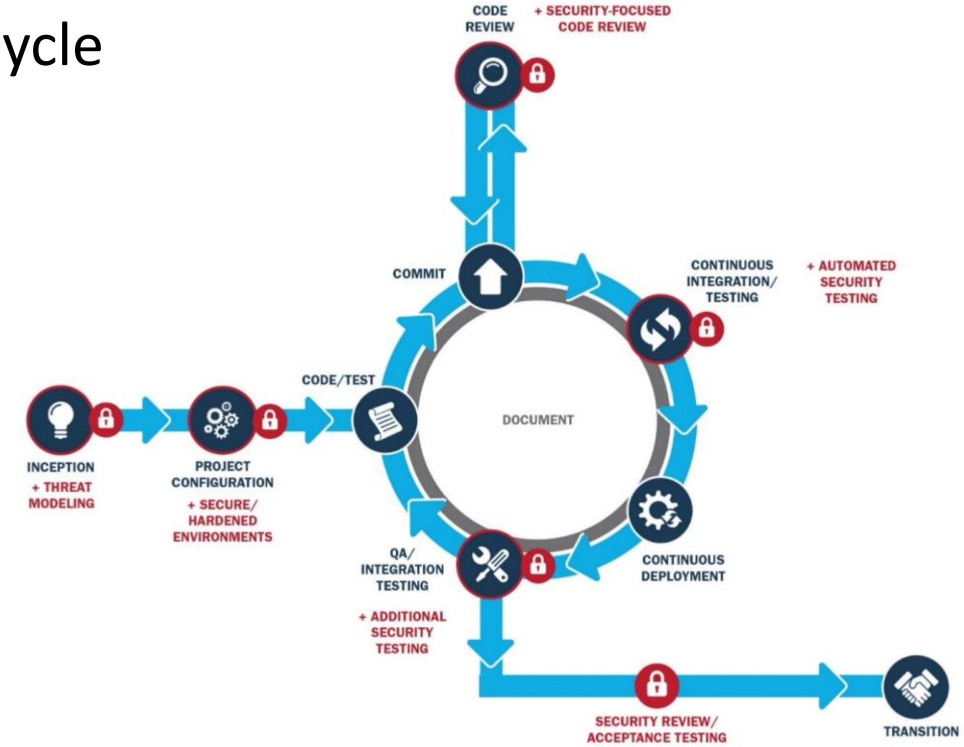


DEVSECOPS

Enhancing SDLC Security

Secure

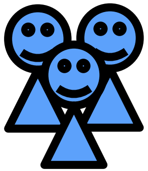
DevOps Lifecycle

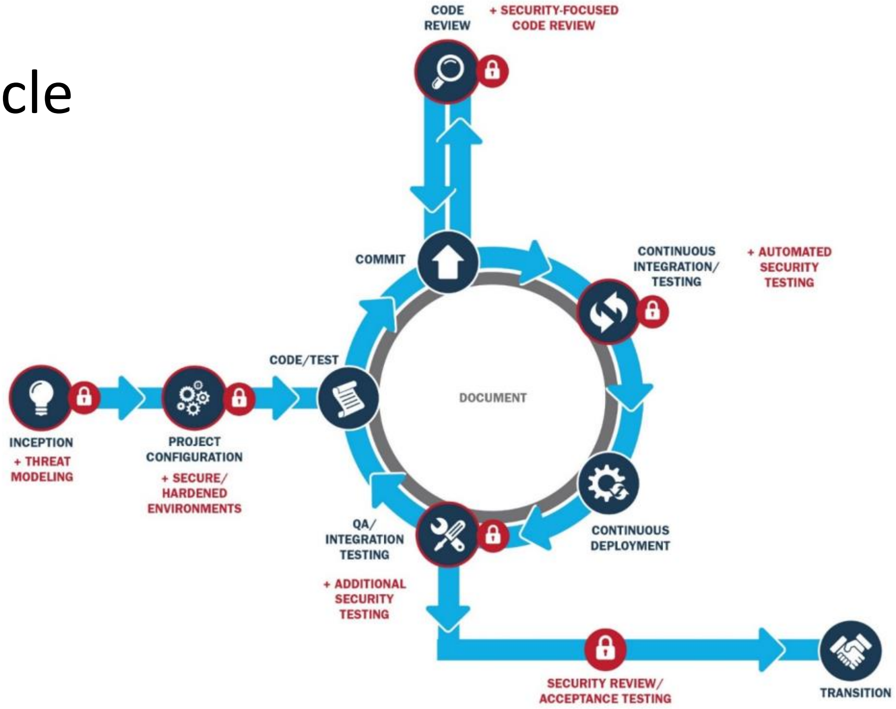

CODE REVIEW
+ SECURITY-FOCUSED CODE REVIEW
COMMIT
CONTINUOUS INTEGRATION/ TESTING
+ AUTOMATED SECURITY TESTING
CODE/TEST
DOCUMENT
INCEPTION
+ THREAT MODELING
PROJECT CONFIGURATION
+ SECURE/ HARDENED ENVIRONMENTS
QA/ INTEGRATION TESTING
+ ADDITIONAL SECURITY TESTING
CONTINUOUS DEPLOYMENT
SECURITY REVIEW/ ACCEPTANCE TESTING
TRANSITION

Security must be addressed without breaking the rapid delivery, continuous feedback model!

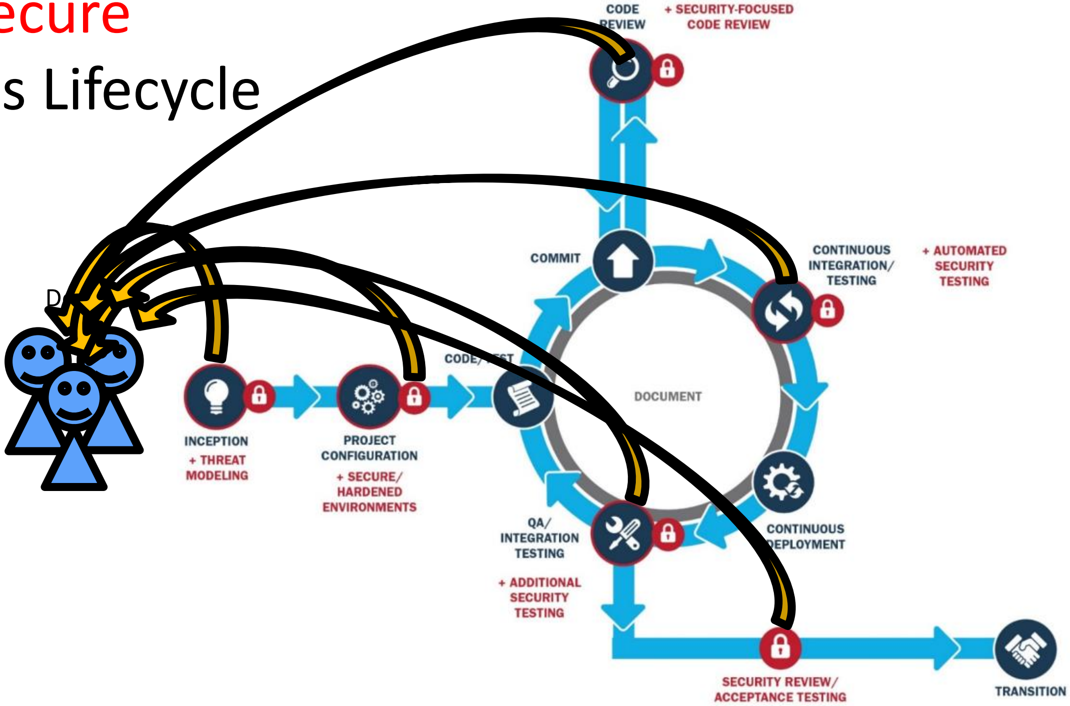
Enhancing SDLC Security

Secure
DevOps Lifecycle




CODE REVIEW
+ SECURITY-FOCUSED CODE REVIEW
COMMIT
CONTINUOUS INTEGRATION/ TESTING
+ AUTOMATED SECURITY TESTING
CODE/TEST
DOCUMENT
INCEPTION
+ THREAT MODELING
PROJECT CONFIGURATION
+ SECURE/ HARDENED ENVIRONMENTS
QA/ INTEGRATION TESTING
+ ADDITIONAL SECURITY TESTING
CONTINUOUS DEPLOYMENT
SECURITY REVIEW/ ACCEPTANCE TESTING
TRANSITION

Enhancing SDLC Security


Secure
DevOps Lifecycle
D
CODE REVIEW
+ SECURITY-FOCUSED CODE REVIEW
COMMIT
CONTINUOUS INTEGRATION/ TESTING
+ AUTOMATED SECURITY TESTING
CODE/TEST
DOCUMENT
INCEPTION
+ THREAT MODELING
PROJECT CONFIGURATION
+ SECURE/ HARDENED ENVIRONMENTS
CONTINUOUS DEPLOYMENT
QA/ INTEGRATION TESTING
+ ADDITIONAL SECURITY TESTING
SECURITY REVIEW/ ACCEPTANCE TESTING
TRANSITION

Continuous Feedback to Developer and **others**

DEVSECOPS

Dos and Don’ts

Automation

- Don’t leave security automation out of your DevOps automation strategy
  - Automated security testing removes human error, infrequent execution, and excuses

✗ Don'ts

- Don’t try to avoid open source with policies, it is coming whether you like it or not!

- InfoSec must maintain awareness of open source vulnerabilities and continuously check for them

Automation

You automate…

✓

Do's

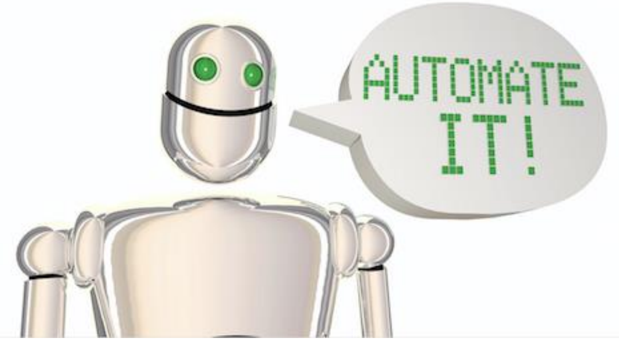
…builds

…functional tests

…deployment

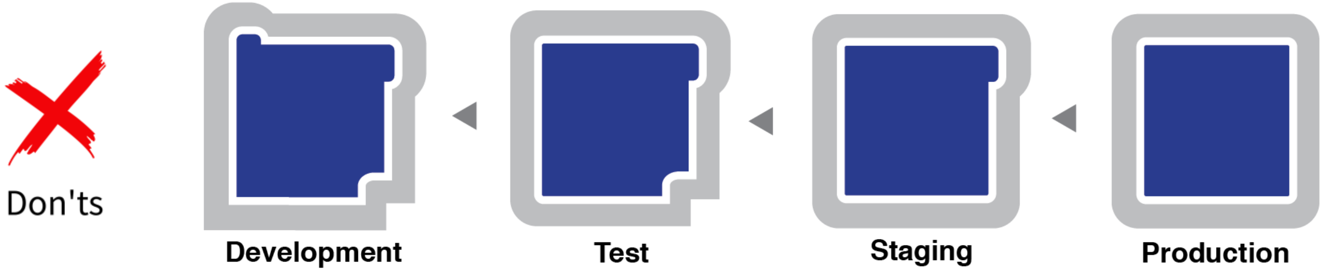
…reporting

…the coffee machine (as we do)



Multiverse: Environment Parity

- When environments are not the same,
  - app may never behave predictably.



- Environment parity (between dev, test, prod) is critical for controlling opportunity for security gaps

Multiverse: Environment Parity

✓

Do's

- Automate manual steps to the extent possible
- Make development environment parity a priority
- Get Ops involved in creating all environments, including Dev
- Focus on providing fast easy-to-use automation tools for ~~developers~~ **everyone** to keep environments in synch

Configuration: IaC

✗

Don'ts

- Uncontrolled configuration changes will lead to an unmanageable, unpredictable, and unrepeatable solution
  - Easy for info security to get out of synch; For example, change in DNS and you have security hole.

✓

Do's

- Avoid the manual quick fix particularly for configuration changes
- Put configuration files under configuration controls

Infiltrator – Insider Threat

✗

Don'ts

- He sneaks in…
- …and alters production …but he works for you!

✓

Do's

- Set up roles and revoke administrative access to manually edit production
- Configure prod environment to alert the entire team when manually accessed. Transparency is key.

Incident:

We have all been there…

Intrusions overnight…

…cascading system failures…

…it's all crashing…

…help…me.....

Response

✗ Don'ts

- But you survive…
  - Glad its over. Going to go sleep for 18 hours…and then back to the normal cycle.
  - When do we analyze what went wrong?
  - How do we prevent similar failures in the future?
  - Just forget it is over!

✓ Do's

- All failures must result in codified change to DevOps process
- Understand exactly what went wrong
- Never let the same failure happen twice
- Propagate fixes across the enterprise
- Ensure that you teach the next generation

Open Source Technology

98% of developers use open source tools (*)

Do you know
what's in your app?

Code we wrote

Code someone else wrote

(*) https://about.gitlab.com/2017/03/03/why-choose-open-source/

Open Source Technology

✗ Don'ts

- Place infosec outside of the dev workflow
- When UI/UX, infosec and accessibility requirements conflict and never get resolved
- Dictate policy to not use open source
- Document-driven checking is not going catch

✓ Do's

- Infosec must enable constant (read: automated) checking for open source vulnerabilities
- Create a centralized private repositories of vetted 3rd party components for all developers
- Establish good product distribution practices
- Minimize variation of components to make things easier (multiple versions, duplicated utility)

- *Prepare for what is coming….*

Continuous Delivery: Rollback

✗

Don'ts

- Once you jump, you can’t return to the plane.
- You are committed. Permanently.
- This is not how we should model our deployments

✓

Do's

- Rollback is essential; Never be left without an escape route to completely working software
- Strive for approaches that support “one button” rollback (e.g, feature flags or A/B)

SLS team GitHub Projects

- Once Click DevOps deployment
  https://github.com/SLS-ALL/devops-microcosm
- Sample app with DevOps Process
  https://github.com/SLS-ALL/flask_api_sample
  - Tagged checkpoints
    - v0.1.0: base Flask project
    - v0.2.0: Vagrant development configuration
    - v0.3.0: Test environment and Fabric deployment
    - v0.4.0: Upstart services, external configuration files
    - v0.5.0: Production environment
- On YouTube:
  https://www.youtube.com/watch?v=5nQlJ-FWA5A

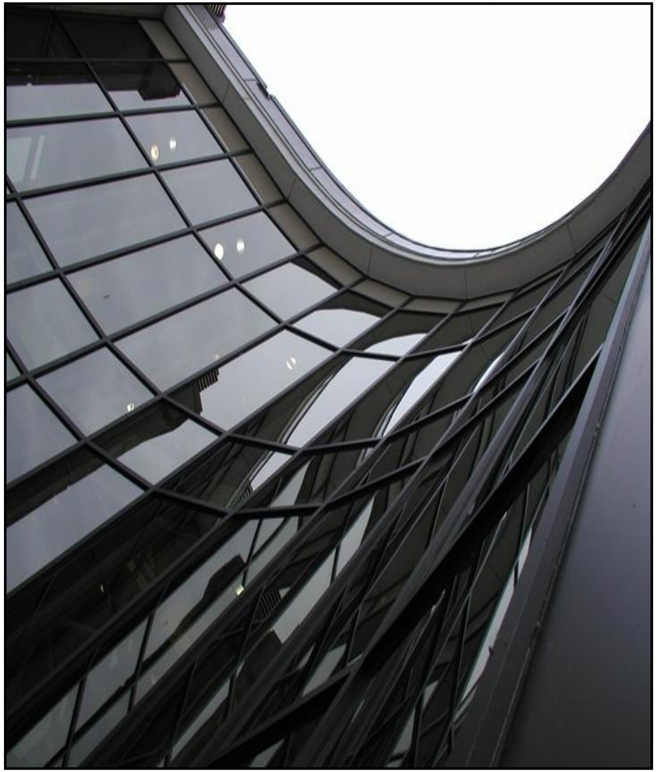
For more information…

- SEI – Carnegie Mellon University
  - DevOps Blog: https://insights.sei.cmu.edu/devops
  - Webinar : https://www.sei.cmu.edu/publications/webinars/index.cfm
  - Podcast : https://www.sei.cmu.edu/publications/podcasts/index.cfm
- DevSecOps: http://www.devsecops.org
- Rugged Software: https://www.ruggedsoftware.org

Let us Apply what we have learned today

- Next week,
  - Change your mindset say “We all are responsible for security” not “You, I or somebody else!”
  - Share what you have learned from failure
- Next Month(s)
  - Start to build Integrated DevOps pipeline
  - Made incremental security integration as part of application lifecycle
  - Measure the results and keep iterating
- By End of 2018!
  - Continuous learning on “how and where we need to improve security of our app”
  - *Use **DevOps** to deploy **secure** application: **DevSecOps***

Any Question?



Hasan Yasar

Technical Manager,
Secure Lifecycle Solutions
hyasar@sei.cmu.edu
@securelifecycle

APPENDIX

DevOps Fundamentals

What is DevOps?

DevOps is a set of principles and practices emphasizing collaboration and communication between software development teams and IT operations staff along with acquirers, suppliers and other stakeholders in the life cycle of a software system [1]

The history of DevOps

- Patrick Debois, “Agile infrastructure and operations: how infra-gile are you?”, Agile 2008
- John Allspaw, “ 10+Deploys per Day: Dev and Ops Cooperation”, Velocity 2009
- DevOpsDays, October 30th 2009, #DevOps term born

[1] IEEE P2675 DevOps Standard for Building Reliable and Secure Systems Including Application Build, Package and Deployment

Who are Dev?

- Follow Agile methodologies
  - Using Scrum, Kanban and modern development approaches
  - Self directing, self managed, self organized
- Using any new technology
  - Each Dev has own development strategy
  - OpenSource,
- Allowed to have
  - Close relationships with the business
  - Software driven economy

Want to deliver software faster with new requirements…

Who are Ops?

- Operations
  - Runs the application
  - Manages the infrastructure
  - Support the applications
- Operations provides
  - Service Strategy
  - Service Design
  - Service Transition
  - Service Operations
  - Secure systems

Want to maintain stability, reliability and security…

DevOps aims to Increase…

…the pace of **innovation**

…**responsiveness** to business needs

…**collaboration**

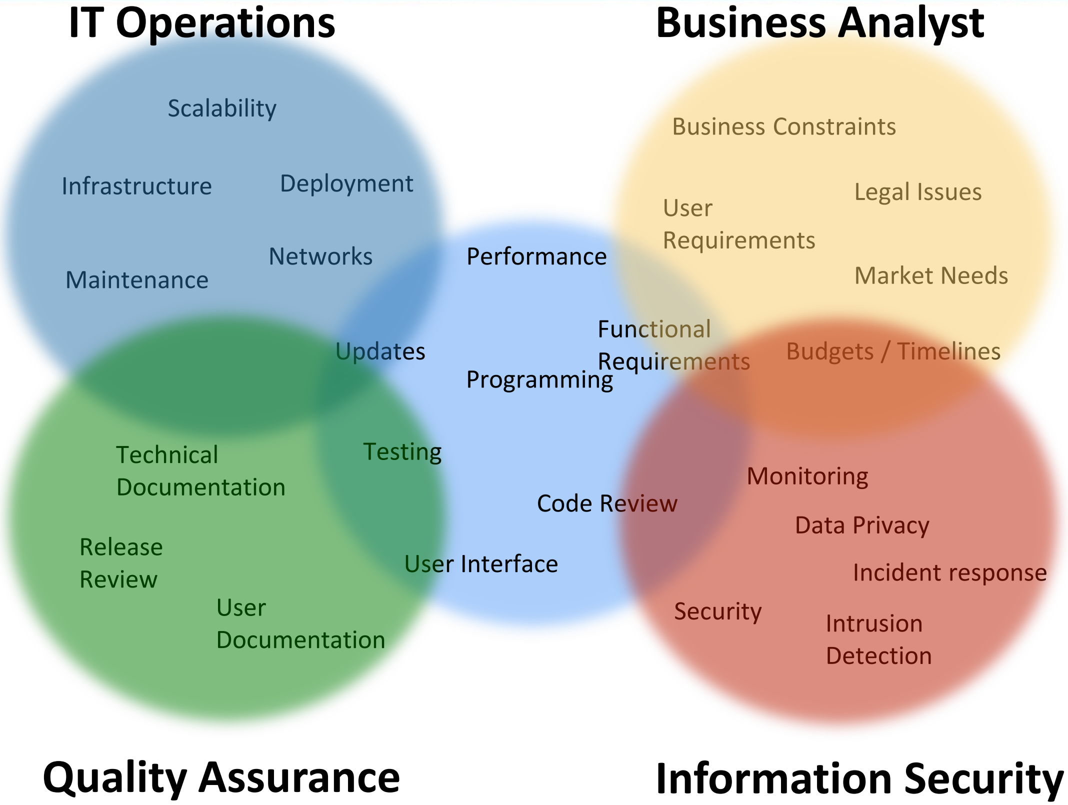
…software **stability and quality**

… continuous feedback

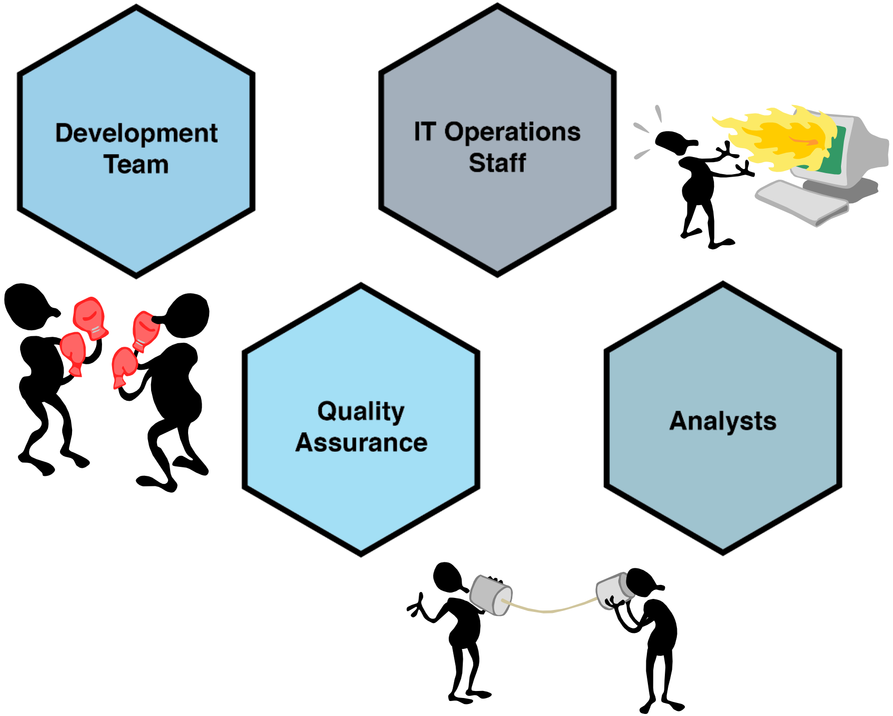
DevOps has four Fundamental Principles

- **Collaboration:** between project team roles
- **Infrastructure as Code:** all assets are versioned, scripted, and shared where possible
- **Automation:** deployment, testing, provisioning, any manual or human-error-prone process
- **Monitoring:** any metric in the development or operational spaces that can inform priorities, direction, and policy

Collaboration: *Many stakeholders*



Collaboration: *Silos Inhibit Collaboration and poor communication*



Development Team

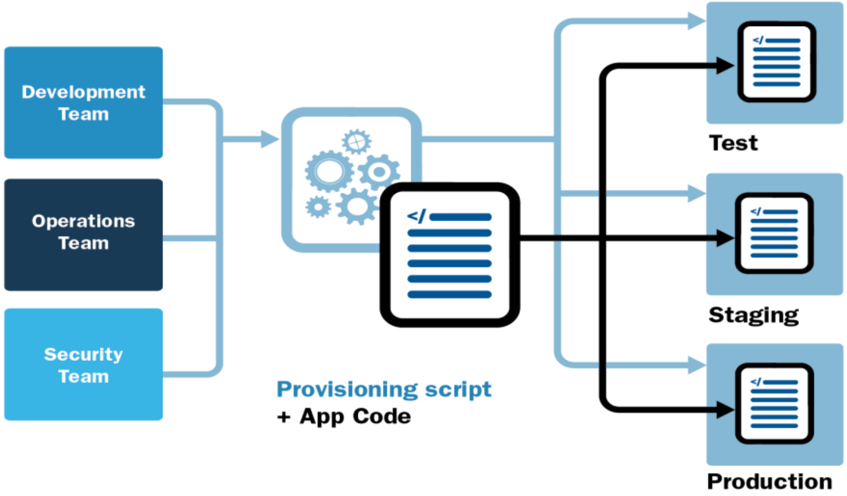
IT Operations Staff

Quality Assurance

Analysts

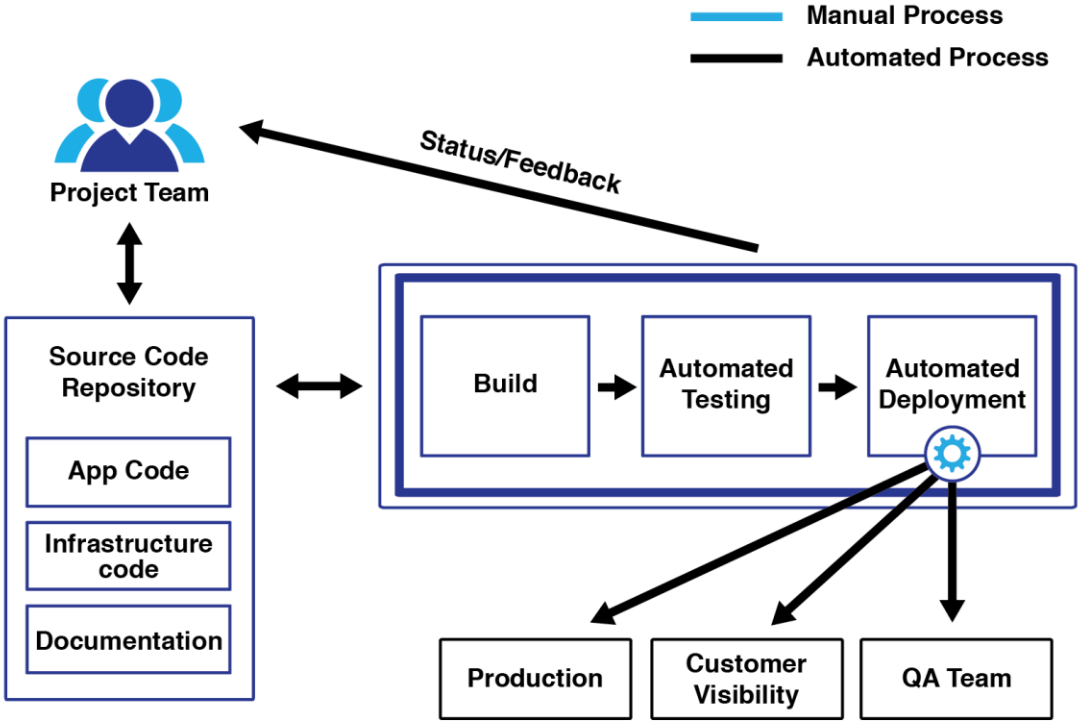
Infrastructure as Code (IaC)

A program that creates infrastructure,

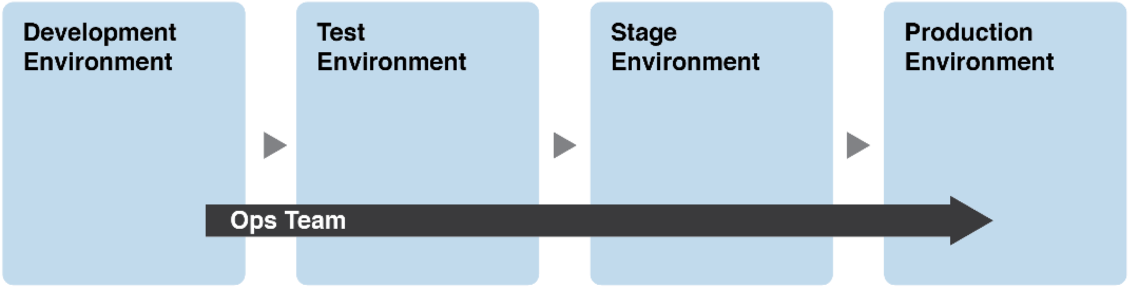


A concretely defined description of the environment is good material for conversation between team members.

Automation : *Continuous Integration (CI)*



Automation : *Continuous Delivery / Deployment (CD)*



Shift Left Operational Concerns Enforced by Continuous Delivery