Automating security testing

THE BRAVE NEW WORLD OF WEB APP TESTING

whoami

- Workday SECENG by day.
- Ph.D. by night.
- @fr1t3 in Twitter.
- Slides/demo @ github.com/frite.

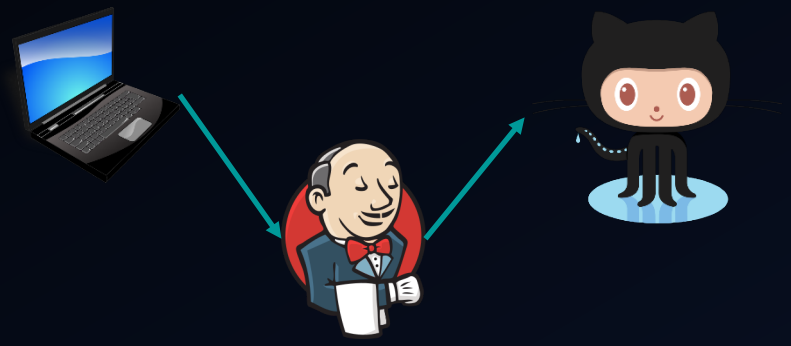
Index

- The current state of web development
  - Developers
  - Dorothy, you are not in Kansas anymore.
  - Security testing and automation
- Current pitfalls
- Automating the testing
  - Demo
- Benefits
- Things to avoid
- Don’t get overexcited.
- Future research

The current state of development

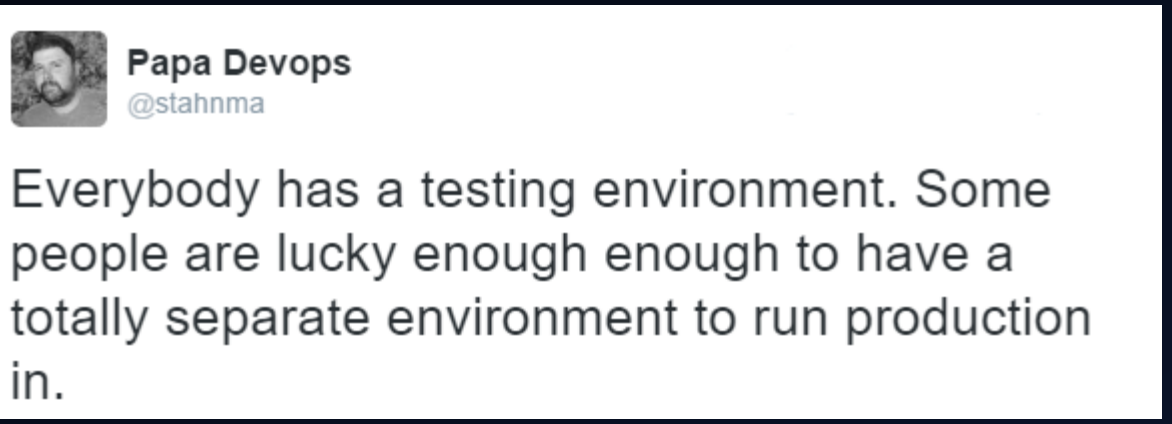
- Products rely on many moving parts.
- Certain parts are built in house.
  - Over time, we end up with many projects.
- Need to ship fast.
- Need to ship parts that are reliable.
  - There are SLAs around this.
  - Calculating 9s etc.
- Technology stacks also change.

Typical Development Workflow (in a nutshell)

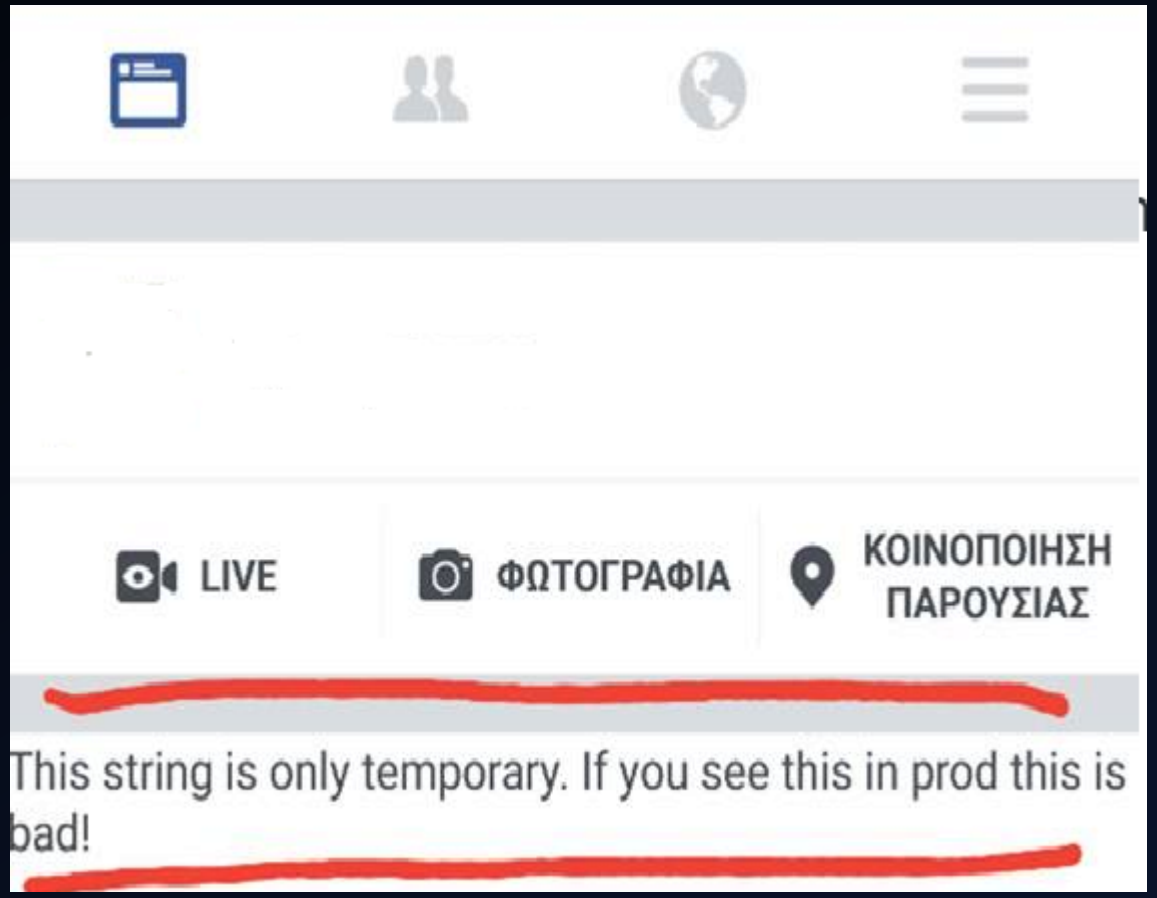


… And ship.

Dorothy, you are not in Kansas anymore.



Dorothy, you are not in Kansas anymore.



Dorothy, you are not in Kansas anymore.

- Developers tend to write tests.
- Typically, development team decides development flow.
  - Test driven, behavior-driven etc.
- They, usually, have:
  - Unit tests.
  - And at some point integration tests.
- They automate the steps above in build servers (Travis/Jenkins).

Security testing

- Static analysis
  - Manual or automated audits.
- Dynamic analysis
  - Either manually or through fuzzing.
- Write our own security tests.
  - i.e. BDD-Security.
- We can train developers to use our tools.
  - It’s not a black art after all.

Current pitfalls

- Projects can easily outgrow your capacity.
  - Lots of repos/code.
- New features ship fast.
- It makes no sense to:
  - Write new BDD-Tests.
  - Pentest every new functionality.
- Let’s exploit what we have.
  - …tests.

Towards security automation.

- All of those projects at some point are merged and tested as a whole.
  - And people hate getting called on a Saturday morning to fix their stuff.
- … thus they write tests.
  - And they also keep their tests up-to-date.
  - They even include new tests for “lessons learned”.
- Web app tests == HTTP requests.
- ZAProxy, Burp etc. are essentially HTTP proxies.
  - …and they provide security functionality.

DEMO

Benefits

- Security teams can focus on other tasks.
- You can actually know if there’s a new issue.
  - i.e. have the build plan to create tickets for new issues.
- Developers also know when they introduce new issues.
- Developers know what matters.

Things to avoid

- Create tickets for every issue.
  - … we wanted to avoid the overload.
  - Issues come with severity.
- Unless you give developers a way to mute specific “issues”, they are going to turn it off.
  - No one likes spam.
  - Also, fine-tuning means accepting their input as well.
- You may want to have this in a separate build plan.
  - You can considerably delay builds. Seriously.
  - You may also want to control the build plan.

Don’t get overexcited.

- This approach **doesn’t** aim to replace the existing approaches.
  - Static and dynamic analysis still needs to be done.
- You still want to sit down with the teams and draft out requirements.
  - #SorryNotSorry.
- In general, don’t put all your eggs in one basket.
  - I.e. use this approach in conjunction with others.
- Finally, you have to spend some time with the team to fine-tune.

Future things

- We can take a similar approach with static analysis.
- We can potentially leverage the same approach with Unit tests
  - And use fuzzing inputs during unit testing.

Q&A

Thanks