BLAZE

INFORMATION SECURITY


BLAZE


TEST DIVE
CONFERENCE'20

Application security testing in the age of Agile development

BLAZE

INTRO

Julio Cesar Fort

Director of Professional Services
at Blaze Information Security

AGENDA

Traditional penetration testing model

Application security engineering

How can QA testers help?

Conclusion

BLAZE

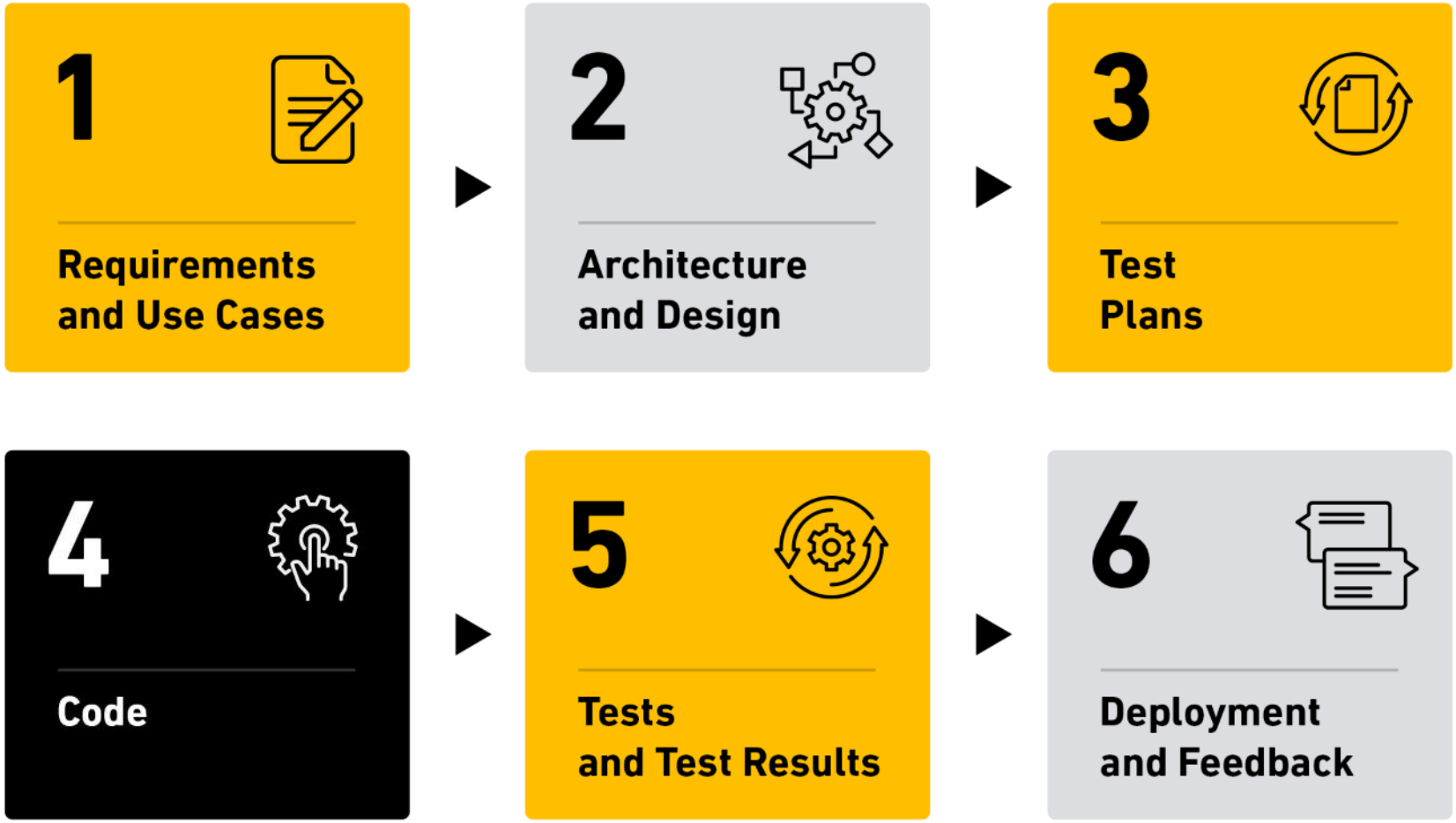
Traditional penetration testing model

BLAZE

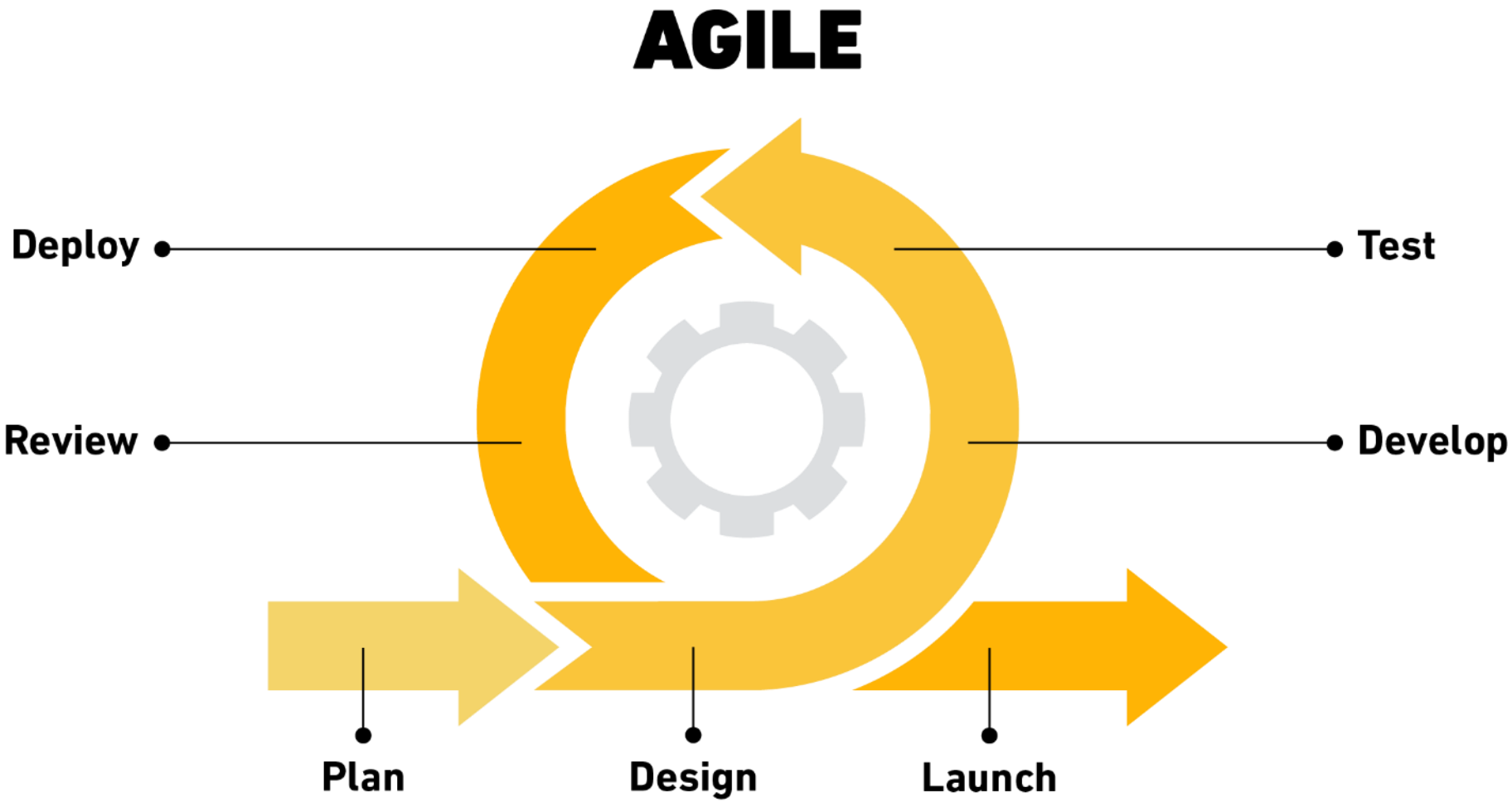
DEVELOPMENT METHODOLOGIES VS. SECURITY

In the past decade or so there was a **shift from waterfall to Agile development** model. We went into Agile / DevOps but **security was once again left out.**

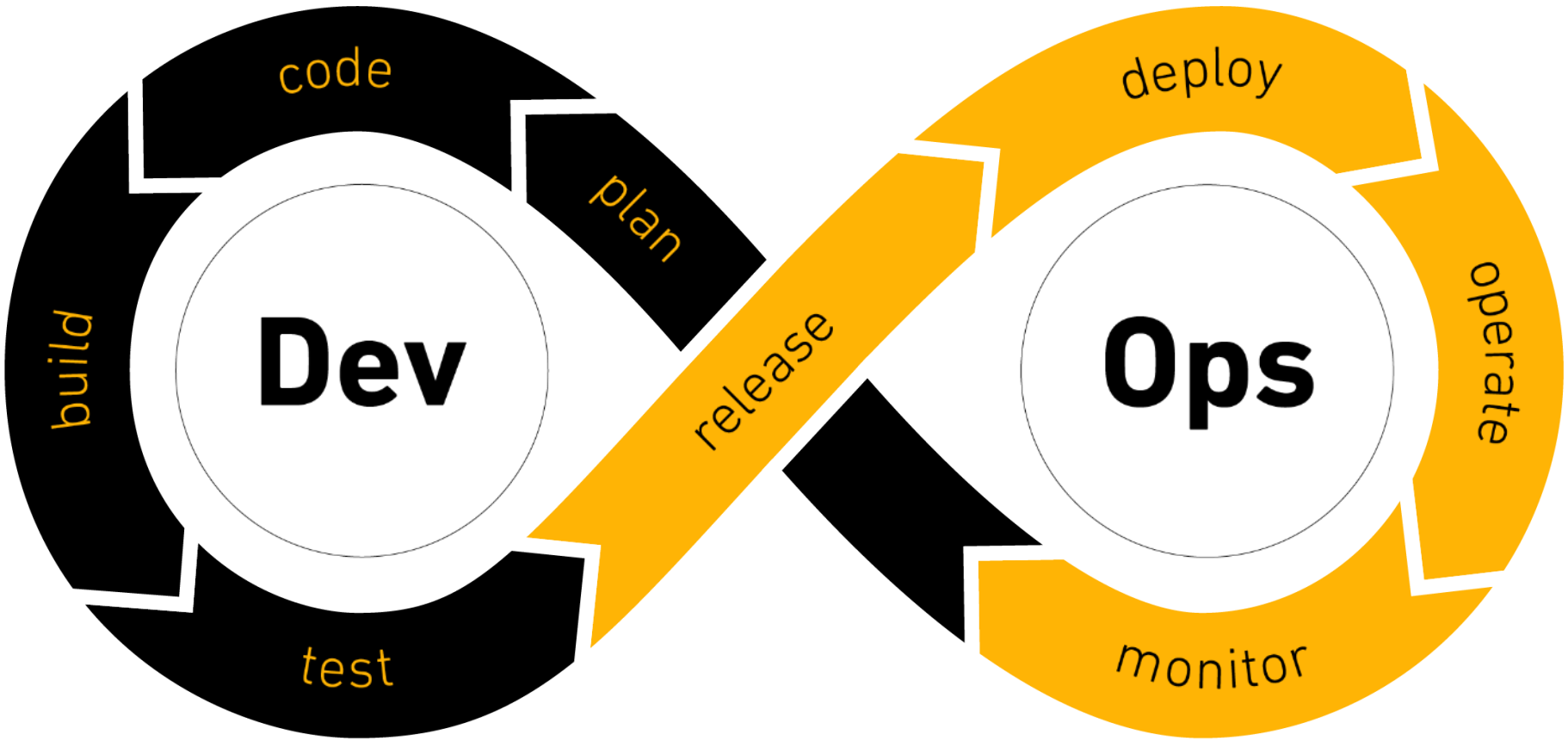
WATERFALL DEVELOPMENT PROCESS



CURRENT SOFTWARE DEVELOPMENT PROCESS



CURRENT SOFTWARE DEVELOPMENT PROCESS



PROBLEMATIC APPROACH TO SECURITY

Think of your system as a cake:
security is often brushed on top of the cake, instead of being **baked in as layers.**

PROBLEMATIC APPROACH TO SECURITY

You push code continuously,
but **why aren't security initiatives continuous too?**

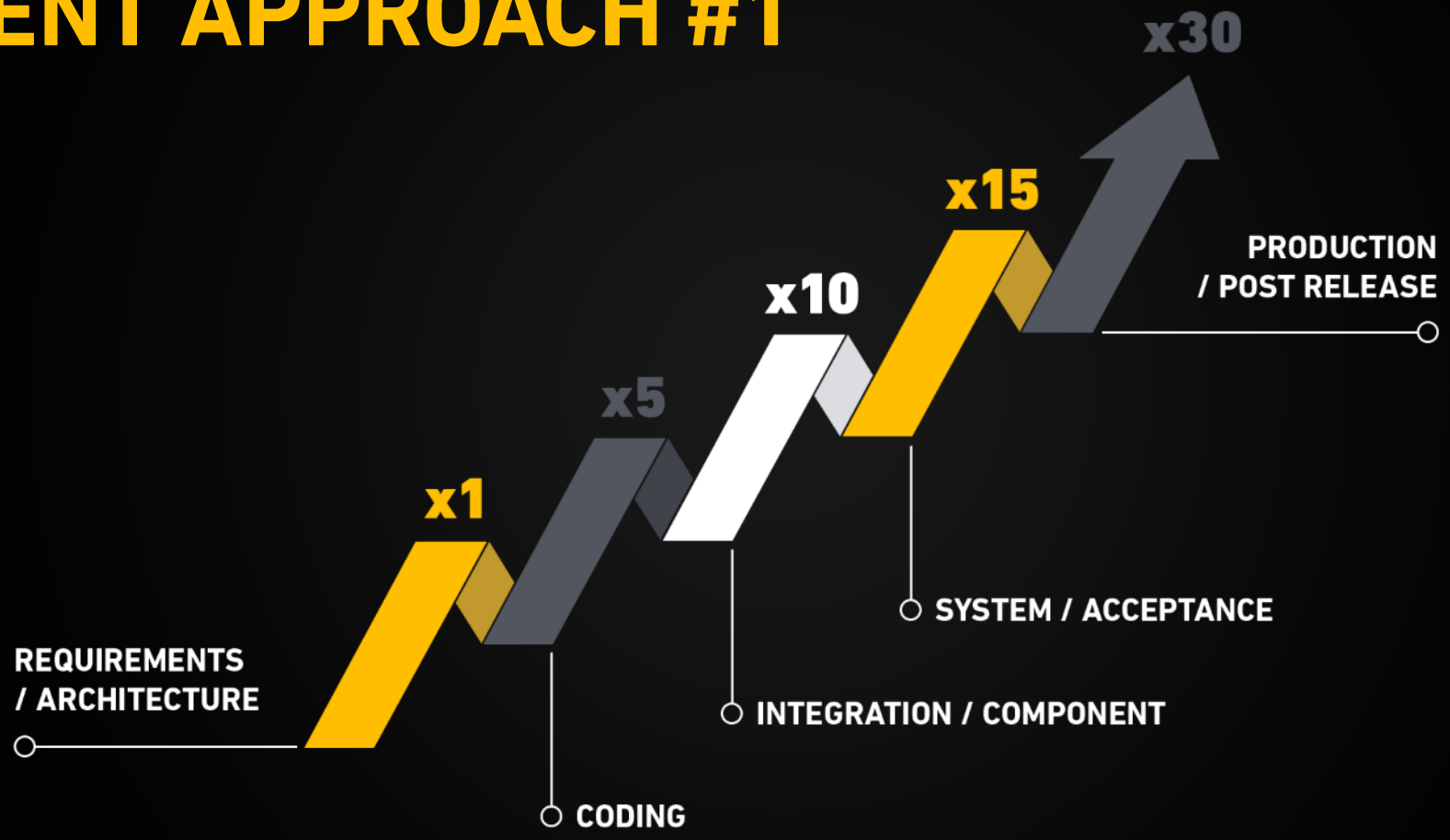
TRADITIONAL SECURITY TESTING

- **Penetrate and patch approach:** has been done since the 60's, with an uptick in the late 90's and the current momentum since 2000's
- **Traditional security testing only occurs right in the end of the process:** after everything has been built, you bring in a team to review / break things
- **Time-boxed engagements:** usually a team gets hired for 1 or 2 weeks for an assessment Impossible to get acquainted with the inner workings of the system and get the best bugs out of it

DISAVANTAGES OF THE CURRENT APPROACH #1

- **Things are broken:** some defects are easier to fix, some others will require more time - delays in the GO-LIVE date
- **Trying to bolt security onto software after it had been developed** produces an insufficiently secure product
- **Cost of fixing issues gradually increases as the end of the development cycle approaches**

DISAVANTAGES OF THE CURRENT APPROACH #1



DISAVANTAGES OF THE CURRENT APPROACH #2

- **Silo culture:** breakers often lack the builders knowledge, edge cases and tricky bugs will be missed
- **Security becomes a bottleneck** for faster releases

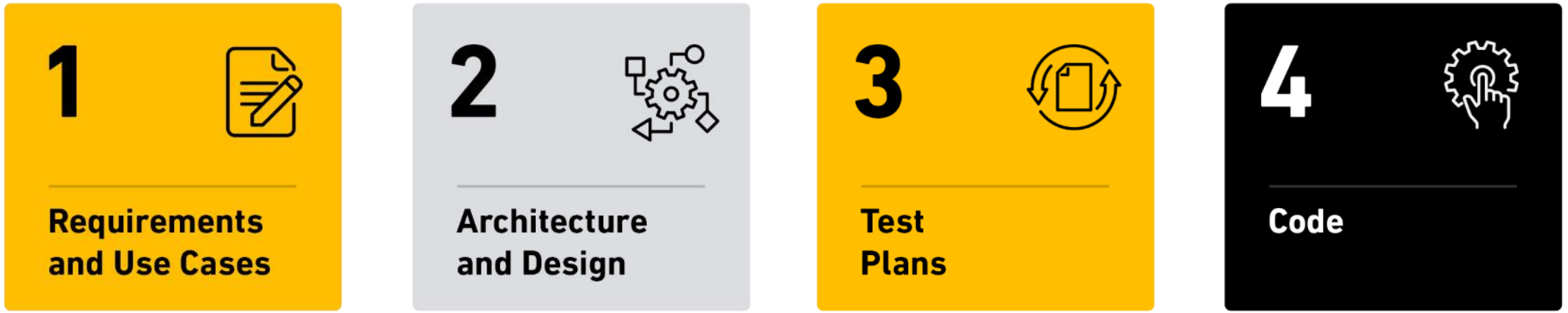
Application security engineering

BLAZE

MODERN SECURITY ENGINEERING IN SOFTWARE DEVELOPMENT

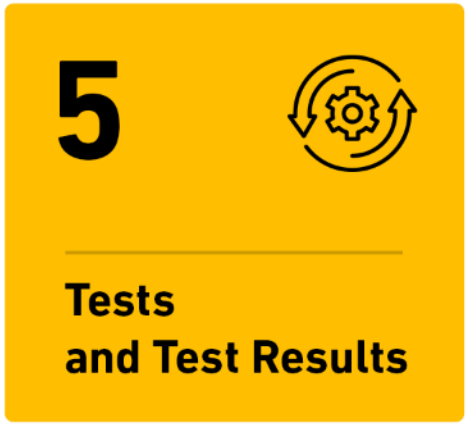
- **Build security in** – since the early days of the project until the day it launches
- **Make security** activities as an **active part** of the project sprint
- Teams: add **“adversarial thinking”** into your user stories

SECURITY ACTIVITIES IN THE DEVELOPMENT PROCESS



Security Requirements Abuse Cases	Risk Analysis Threat Modeling	Introduce Security Oriented Tests	Security Code Review

SECURITY ACTIVITIES
IN THE DEVELOPMENT PROCESS



Vulnerability Scanning
Penetration Testing

SECURITY ACTIVITIES IN THE DEVELOPMENT PROCESS

SecOPS
Continuous Penetration Testing

MODERN SECURITY ENGINEERING IN SOFTWARE DEVELOPMENT: IN THE IDEAL WORLD

Developer training on the subject of cyber-security

General security consultancy on best practices and security review of all requirements

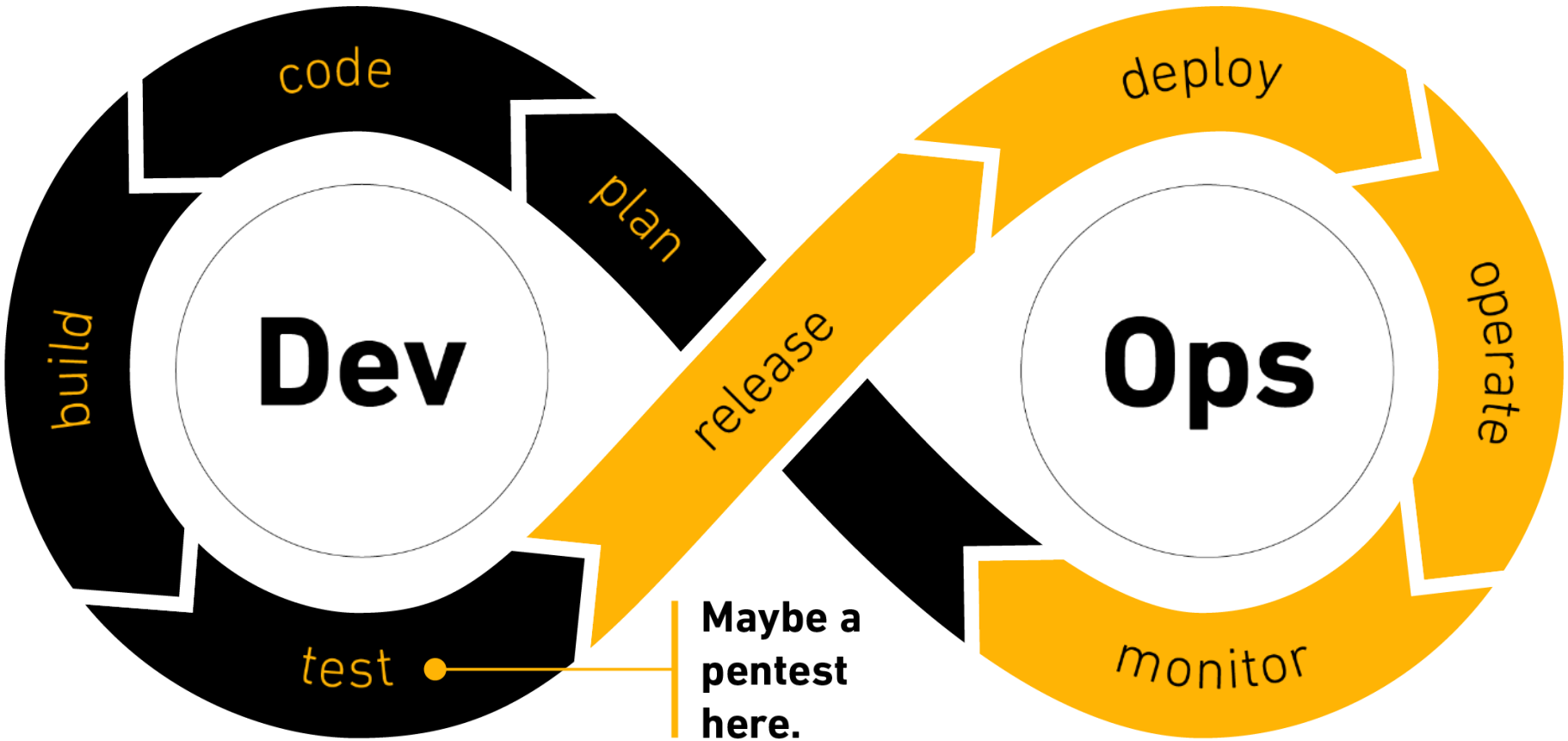
Threat modeling activities in each sprint

Source code review of code pushed out on a regular basis

DevSecOps with open-source and other proprietary tools

Penetration testing **in every major release**

AND IN THE REAL WORLD...



How can QA testers help?

BLAZE

TOWARDS SOFTWARE ASSURANCE ON A SHOESTRING

What can we do as QA's to help an average development team with little resources to improve **security maturity?**

TOWARDS SOFTWARE ASSURANCE ON A SHOESTRING

- **Ops team:** add **open-source security controls** into your DevOps pipeline
- **Security team**: teach QA's application security basics and how to identify easy bugs

QUICK WIN #1: CROSS-SITE SCRIPTING

- **Cross-site scripting (XSS)** is an attack that can compromise users of a web site
- Often used to **execute scripts** on the user's browser
- One of the **most common** class of vulnerability affecting web applications
- **Sample payload – how to test:** "><script>alert(1)</script><"

QUICK WIN #2: TEMPLATE INJECTION

- **Server-Side Template Injection (SSTI)** happens when user input is injected into a template and rendered by a template engine
- Template rendering is **frequently executed in server-side**
- **Sample payload – how to test:** {{7*7}}

QUICK WIN #3: IDOR

- **Insecure Direct Object Reference (IDOR)** occurs when user input is used to access objects directly
- IDORs are a common **access control problem**
- This type of issue is **usually easy to find**
- **How to test:** Seen ?id=1 - so why not cycle through id=2, id=3, id=4....

QUICK WIN #4: SERVER-SIDE REQUEST FORGERY

- **SSRF** allows an attacker to instruct the application to issue requests on his/her behalf
- It can be used to **connect to resources located in the organization's internal network**, interacting with otherwise inaccessible back-end systems
- The result of a successful SSRF can range from **leaking cloud secret keys and even code execution**
- **How to test:** A parameter taking a URL? Try https://www.myip.is . If the IP isn't yours but the server's, you may have a SSRF

QUICK WIN #5: MISCONFIGURED AUTHORIZATION

- The idea of authorization is to **allow or restrict access to a given resource** to a user
- Horizontal vs. Vertical **privilege escalations**
- **How to test:** get two sessions from different users (A and B). Try to use A's session tokens to access the same resources expected only to be available for B

Conclusion

BLAZE

CONCLUSION

- **Security is not always that complicated** and thinking this way makes it harder
- Code is being **pushed into production faster** than ever before, and security did not catch up – but this is changing

CONCLUSION

Introducing security into the DevOps pipeline and QA even on a budget **may yield good results** if your team can't afford dedicated specialists.

CONCLUSION

A lot of application security testing is just **glorified QA** with different tools and hacker-sounding names. Really.

CONCLUSION

With Agile,
security is now part of everyone's job.

Questions?

BLAZE

Be secure. Be ahead. Be Blaze.

THANK YOU!

www.blazeinfosec.com

info@blazeinfosec.com

Brazil
Rua Visconde de Jequitinhonha 279
Office 701, Recife

Portugal
Praça Bom Sucesso 131
Península, Office 206, Porto

Poland
ul. Józefa Sarego 5/4
31-047, Kraków

BR: +55 81 3071 7148
PT: +351 222 081 647
PL: +48 792 436 755

BLAZE