As We Far-Enough to the Left Yet? DevSecOps, Chaos Security, and Not Being Left Behind

Eddie Schwartz

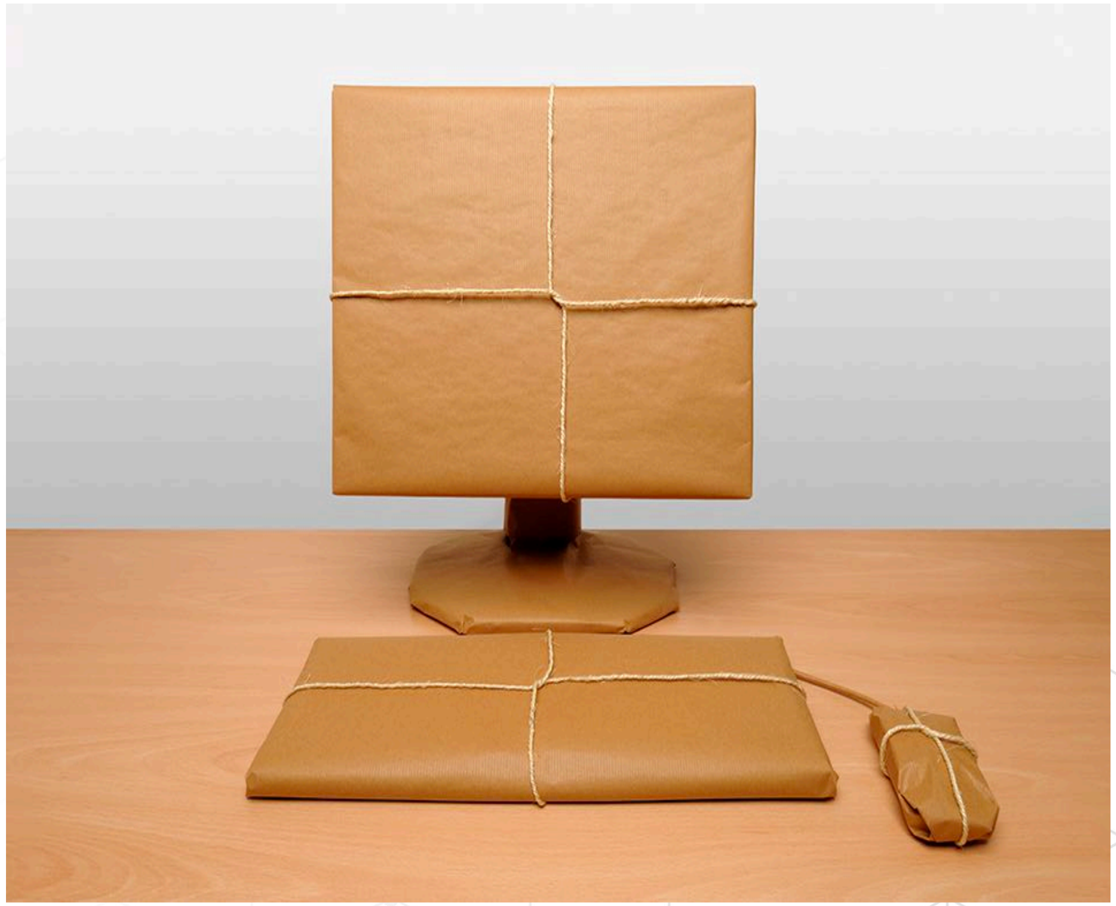
Chief Security Officer, Block.one

This presentation contains my personal views and opinions and not those of my employer.

HITB LOCKDOWN 002 livestream

Let’s Talk About Product Security

- How We Used To Do It
- How We Are <u>Still</u> Doing It…Mostly
- How IT is Doing It…Without Us
- “Moving More to the Left” or How We Could Do it Better
- Q&A



THE GOOD OLD DAYS
DYSTOPIA
ASDA MCE

Product Security Wasn’t What You Think…

- In the early days…The Rainbow Series and TPEP
  - First, you needed to know how the system would be used and what it would be doing
  - Then, you needed to know info such as who would use it and where it might be used
  - Then it was all about testing, evaluation, and proving what you thought was true, is true
  - The more sensitive and critical, the longer and harder it was

The Long and Winding Road to Secure SDLC

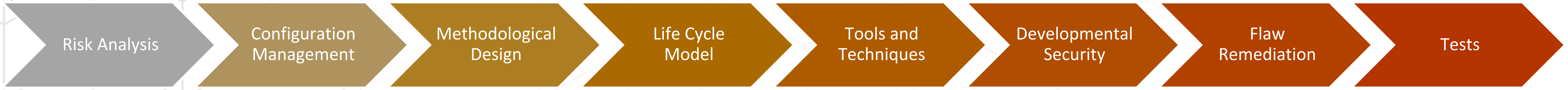
NIAP

Common Criteria (CCEVS)

NVLAP

PPs and TOEs

Common Criteria (CC) Secure Software Development Approach (ISO 15408)



- This approach included additional new aspects to security by design and functional security integration that previous generations didn't have including:
  - Threat Modeling and Security Architecture
  - Secure Design and Requirements
  - Other models emerged that included these dimensions and other tangents

What We Learned from Doing the Same Thing for So Long: Key Elements of Security By Design

Risk Analysis	Identify the most probable threats and analyzing the related vulnerabilities	Secure Configuration Management	Includes secure access to source code, build pipelines and other SDLC artifacts
Threat Modeling	Understanding “what can go wrong” by understanding meaningful security risk	Physical Security and OPSEC	Security for the physical operating environment including issues with users and admins
Security Requirements	Techniques, methods and standards for reducing identified risks	Vulnerability Management	Black box and white box testing. Management of inherited and new vulnerabilities
Security Architecture	The secure aspects and standards of the design of the system	Secure Integration	Integrating COTS and XaaS capabilities into your code securely
Code Analysis	Various techniques to review the code, peer review, static, dynamic, fuzzing, etc.	Security Training	Helping all developers be part of the security team and have fewer security bugs

The Good and the Bad of the Old Ways

- The Good:
  - Understanding WHAT you planned to build
  - Having defined security architecture, requirements and controls
  - Having a clear testing regime that could prove requirements were implemented

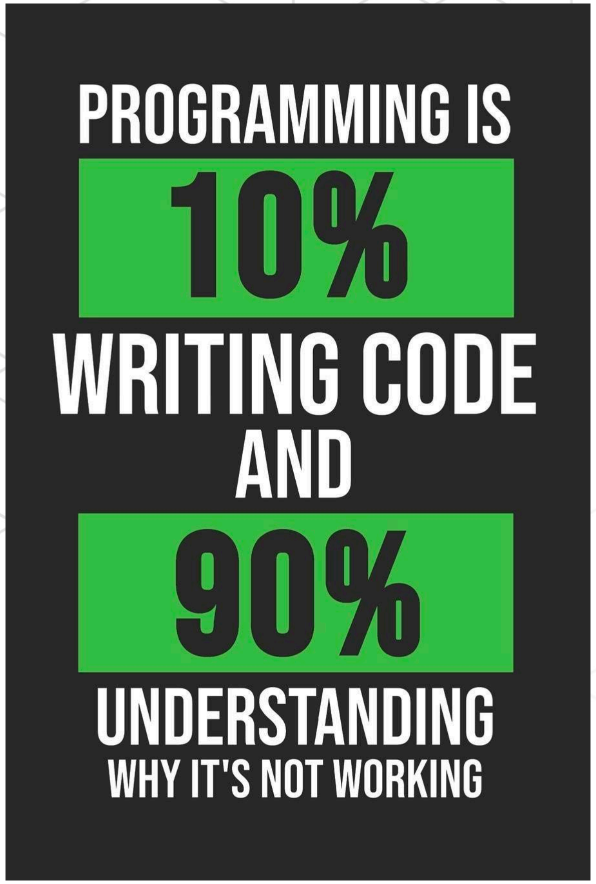
- The Bad:
  - The threat modeling was not typically dynamic
  - TPEP took FOREVER for high security systems, CC and other models not much better
  - Threats and tech change too often, so this approach is not practical for volatility and DevSecOps

Can you play that song again?

So, the Old Approach Seems OK, Right?

- No, it's not working and needs to change, but:
  - It's HARD to do product security
  - It involves CODE, and most of us in security weren't trained for that – the ones who were don't want to do it that way anymore, etc.
  - It's not a pen test
  - It's not a list of NIST or ISO 27001 controls
  - It's not a compliance checklist
  - The security team can't do it in a silo and need to work with others
  - IT / Engineering won't just do what we say because they are more aligned with the business than we are

What are they doing over there in IT?

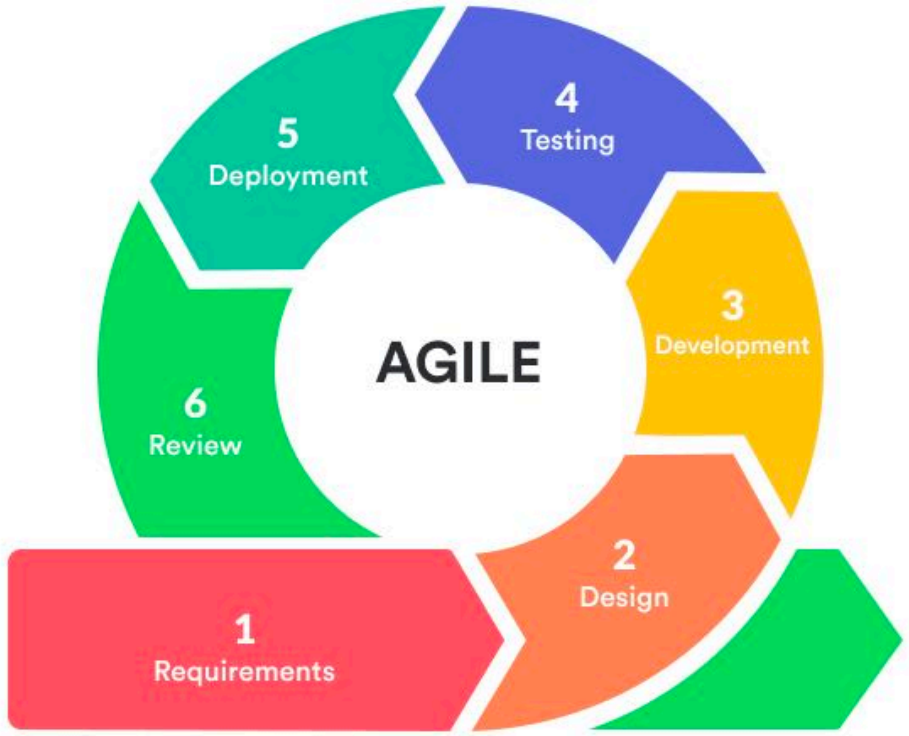


- They are busy over there in IT, Engineering or whatever you call it….
  - They have a "product backlog" and they are performing two-week sprints
  - They are doing peer reviews and "retrospectives"
  - They are constantly updating and changing things
  - Two weeks? I mean, come on, really?
- Meanwhile, we in security are waiting to do our pen tests, trying to figure out what Kubernetes security configurations should look like or what "standards" we should provide for Python programmers…

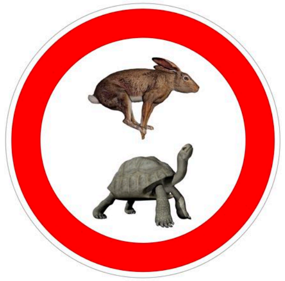
HELP

Engineering Really Does Want Security Help!

- Nobody wants some security or privacy-related mess on their hands
- But Engineering and IT move at a certain speed / cadence – they can't wait for Security to give their blessing
- They want a partner and need a Security SME
- They want us to work the way they do versus follow a "security framework"



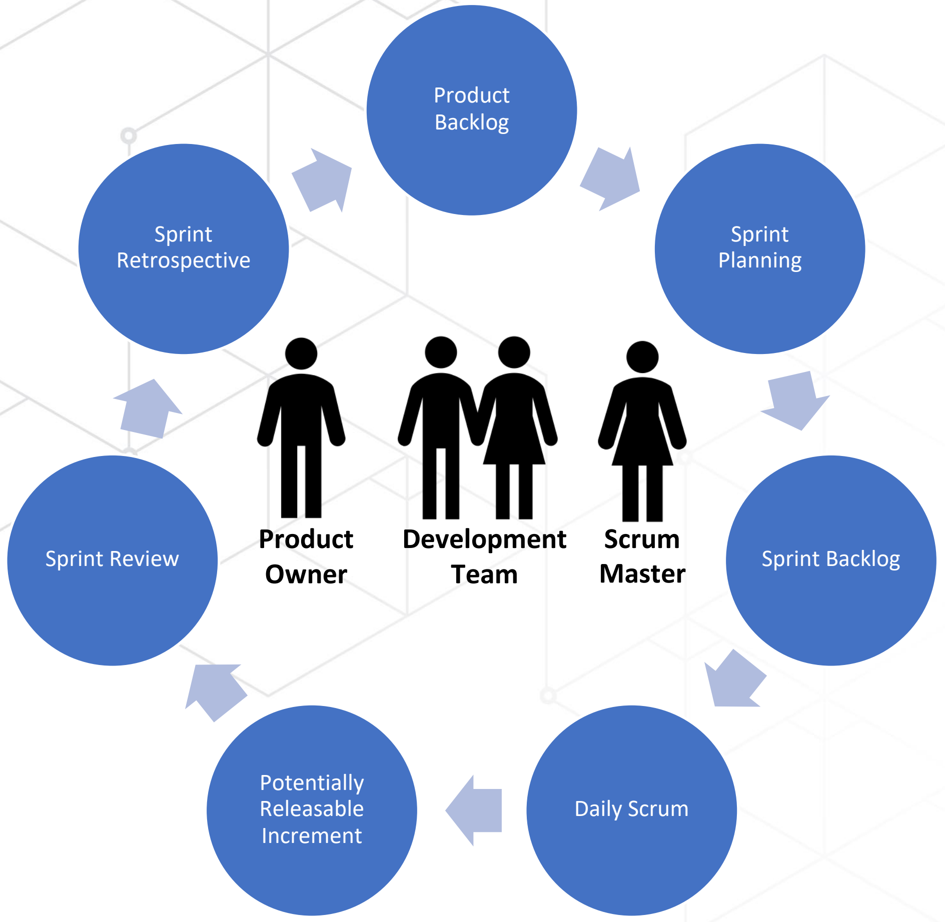
What Are Some Key Differences?



- Engineering
  - Iterative
  - Encourages interaction
  - Transparent
  - Promotes frequent adjustments
  - Delivers working products quickly (Beta, MVP, etc.)
  - Comfortable with uncertainty

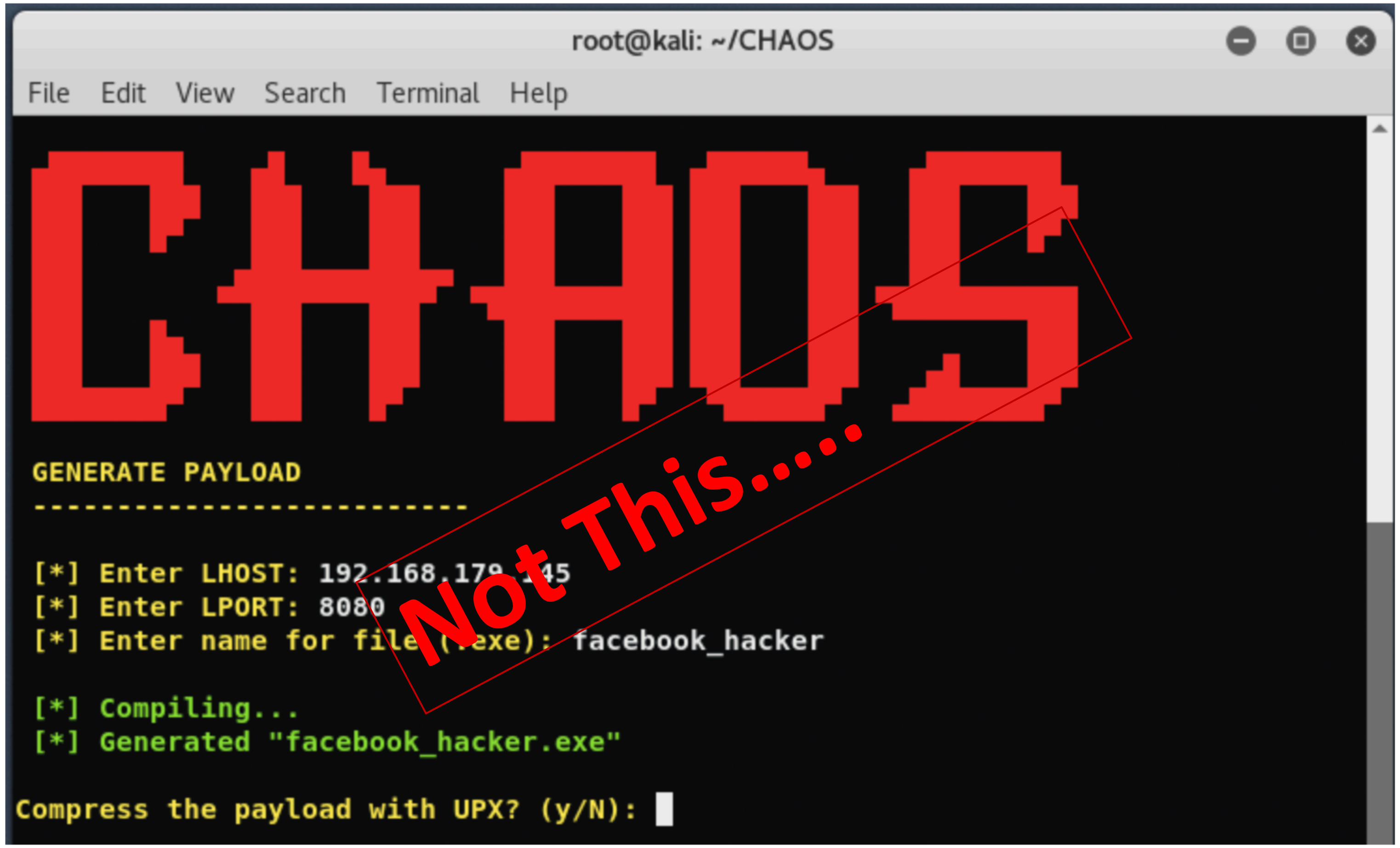
- Security
  - Point in time
  - Top down dictated
  - Secretive or siloed
  - Dislikes frequent changes / pivots
  - Struggles with short timelines or release of incomplete products
  - Uncertainty is bad

IT Keeps Moving Faster and Is Becoming More Agile



- Security doesn’t have a natural home on this slide the way we think about it today. For example:
  - A security “feature” like MFA has to be part of the Product Backlog to make it into Spring Planning, etc.
  - If security requirements are not clear from the architecture or design of the software, the requirements that have to be written into user stories or be part of the natural capabilities of the team or its cycle

And They Are Using CHAOS Too…A Different One



```
GENERATE PAYLOAD
------------------------

[*] Enter LHOST: 192.168.179.145
[*] Enter LPORT: 8080
[*] Enter name for file (.exe): facebook_hacker

[*] Compiling...
[*] Generated "facebook_hacker.exe"

Compress the payload with UPX? (y/N):
```

- In fast-paced environment, there isn't enough time for QA testing, and it has diminishing returns
- Chaos Engineering is the use of experimental and potentially destructive failure or fault injection testing to uncover vulnerabilities and weaknesses within a complex system
- It provides an approach allows for organizational learning, increased reliability and a greater understanding of complex system dependencies

New Life
Old Life

DevSecOps – What is It?

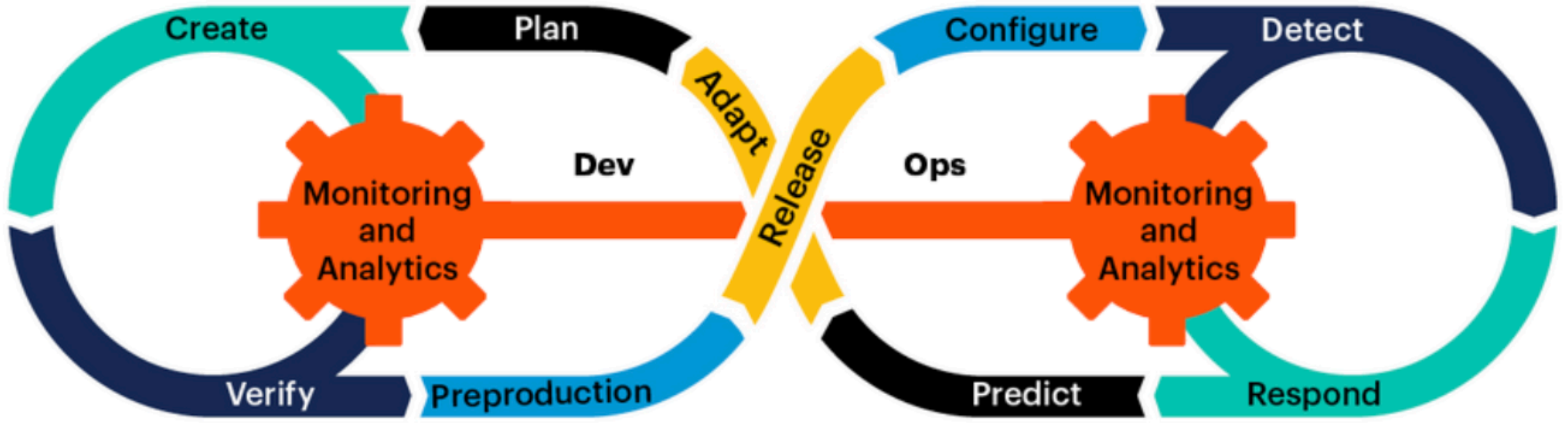
- DevSecOps is the complete integration of traditionally siloed security functions into a DevOps or Agile engineering environment
- Differences between organizations that are "agile" or DevOps often revolve around the wall that may exist between development and operations or product security and security operations
- Depending upon your company, any hybrid of these approaches may be adopted and evolved as your engineering requirements change
- But the key common element is full integration between security and engineering

Illustrative Activities within DevSecOps…Let's Focus on the Left for Now….

- Incremental Threat Modeling
- Secure Design and Architecture

- CM
- Network Security
- Cloud, Kubernetes, Container Security

- Log Management and Analytics
- Security Monitoring
- SOC / Threat Hunting
- Vulnerability Management



- Software Comp Analysis
- Code Review and Testing
- Cataloging of vulnerabilities and risks

- Automated Testing
- Container and Application Signing
- Pipeline Security

- Compliance Management
- Incident Response

HITB LOCKDOWN 002 livestream

DevSecOps: Plan and Create

- For a new product or service, an initial product definition and design will be created by engineering and stakeholders
- Security works as part of the team to create the design of the product and the overall architecture
- The Plan and Create phases become 2- or 4-week sprints where product "increments" are created iteratively
- NOTE: It's critical that the ProdSec team members be part of the Scrum team, have the right technical skills, and be available for the sprint cycle activities

- Deliverables:
  - Threat Model (that will become an incremental threat model in subsequent iterations of plan and create)
  - Security Architecture – including data flow diagrams
  - Security Themes and Objectives
  - Security Requirements
- These deliverables are iteratively updated by the entire team are transparent to the team in JIRA, Confluence or a similar system
- Threat modeling may be tracked in a variety of commercial tools and there are open source threat modeling as code tools

DevSecOps: Verify

- The Verify phase is where security tools and techniques are applied to the work performed during the Sprint by the Development Team
- Additionally, SME security support may aid in both security tool operation or code review during this phase
- Ideally, secure coding practices should be fostered within the Development Team (with help from Security)

- Deliverables:
  - Ongoing results and improvements from SCA tools
  - Analysis and fixes applied from continuous application of SAST and DAST
  - Peer manual code review results as required based on threat model-driven needs
  - Issues revert to product backlog as needed and are tracked in JIRA (or other tool)

DevSecOps: Preproduction and Release

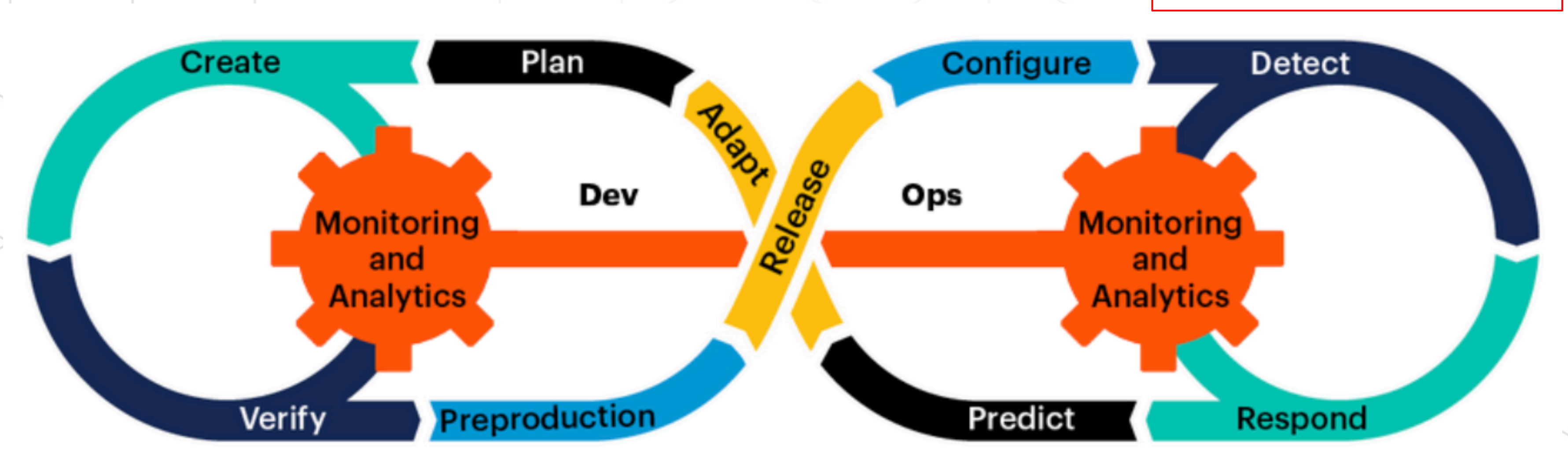
- CI/CD requires automation and removing as many manual processes as possible – security too
- Most preproduction and release processes should be automated and only should hold or stop when “high” or “critical” error conditions are met
- This aspect of DevSecOps requires dedicated focus and deep technical expertise from security teams because it is both error-prone and complex

- Deliverables:
  - Automated pipeline testing including any final SAST and DAST
  - Cryptographic signing of various components of the increment by designated authorities
  - Integrity verification of the pipeline process itself as part of the release process

Some Guidance for the Right Side of the Equation

- CM
- Network Security
- Cloud, Kubernetes, Container Security

- Log Management and Analytics
- Security Monitoring
- SOC / Threat Hunting
- Vulnerability Management



- Compliance Management
- Incident Response

HITB LOCKDOWN 002 livestream

General Right-side Considerations

- If you are in a DevOps shop and using containers, your infrastructure is likely deployed as code – this is good
- You can deploy all kinds of fun things as code, for example: secure configurations, compliance auditing, vulnerability management and other security monitoring agents
- Your security team must be embedded with the engineering team that performs pipeline automation to achieve security goals
- <u>Key takeaway</u>: Many of the product backlog items that occur on the right side of DevSecOps are internal product capabilities that the product managers and owners may not consider on their own
- It's critical for the security team to ask for internal product backlog items that satisfy critical needs for security operations the right side
- Ideally, these backlog items would be treated like any other backlog item, and be executed by the development team (automation team, for example) within a sprint
- However, depending on your individual situation, you may need a security-focused development team for internal-facing security backlog items of this type, particularly if they are large in nature

Thank You!

HITB LOCK DOWN 002 livestream

Eddie Schwartz, @eddieschwartz