HERDING CATS IN A DEVSECOPS WORLD

ROB CAMPBELL – ENTERPRISE SECURITY ARCHITECT

EMAIL – COSAC@ASSUREDCONTROL.COM

V2.0 16 SEPT 2021

Quote - "Currently, DevOps is more like a philosophical movement, not yet a precise collection of practices, descriptive or prescriptive"

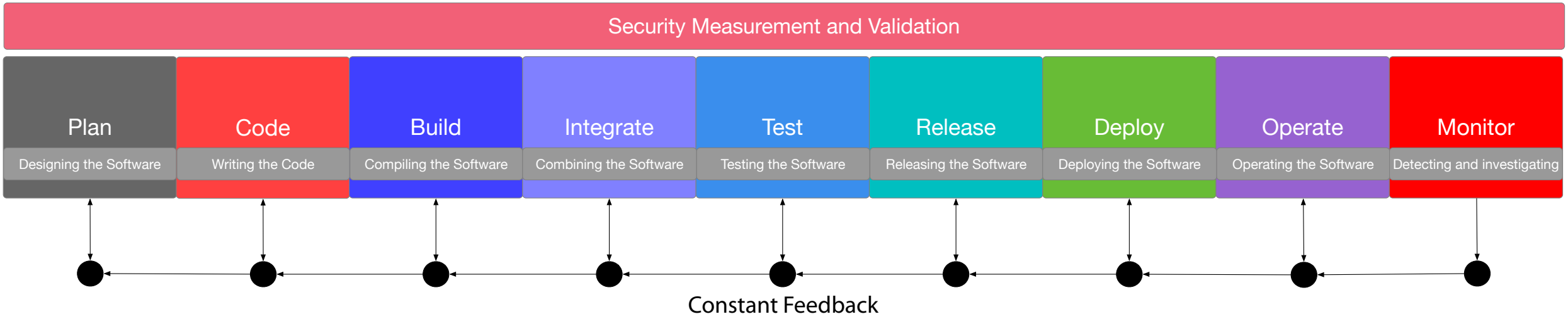
BACKGROUND

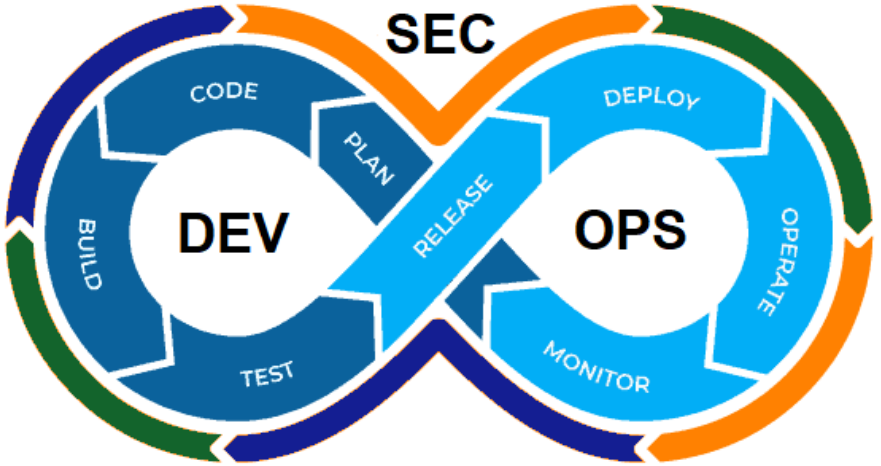
- Application Security is different
- The development Process brings unique threats and challenges
- Developers want to work without barriers
- Vulnerabilities can pop up at any time
- Code theft (I mean reuse) is rife
- Delivery speed challenges traditional ways of working
- Deployment models have changed

AGENDA

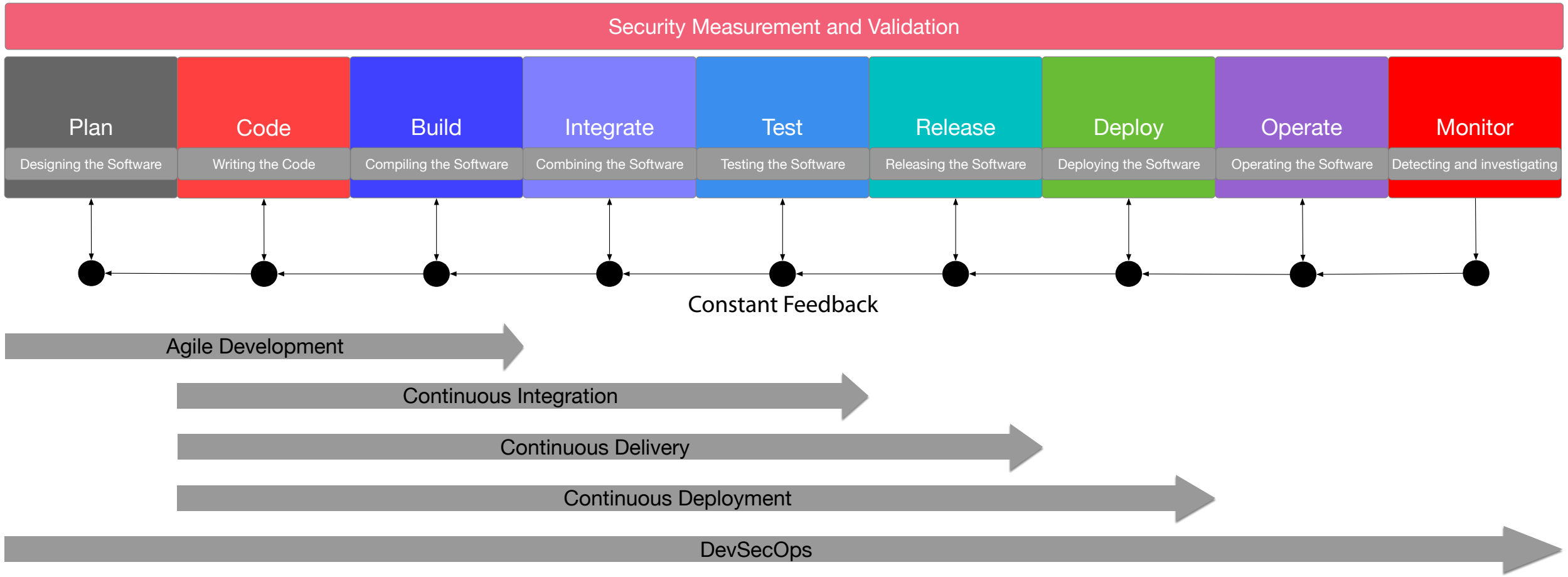
- What is a Devsecops pipeline?
- How the terminology fits?
- What Components make up a pipeline?
- What are the Security concerns?
- What tools are in our kitbag?
- What are the controls?

What is a DevSecOps Pipeline



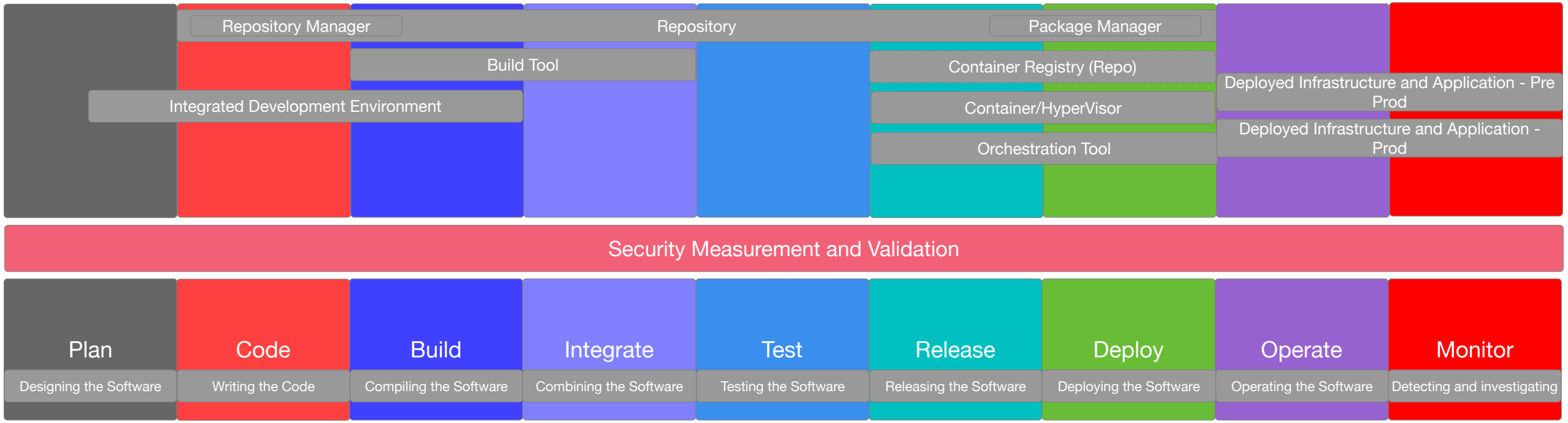


How the DevOps Terminology Fits



What are the components within a Pipeline?

This applies to your Application Development pipeline and your Infrastructure as Code pipeline

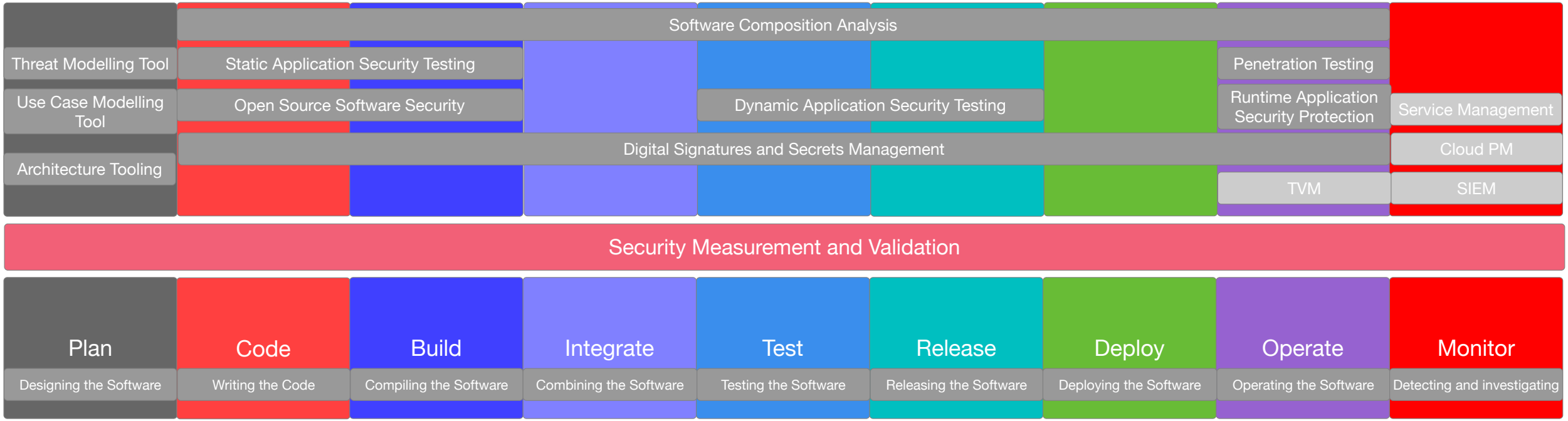


So what are the Security Concerns?

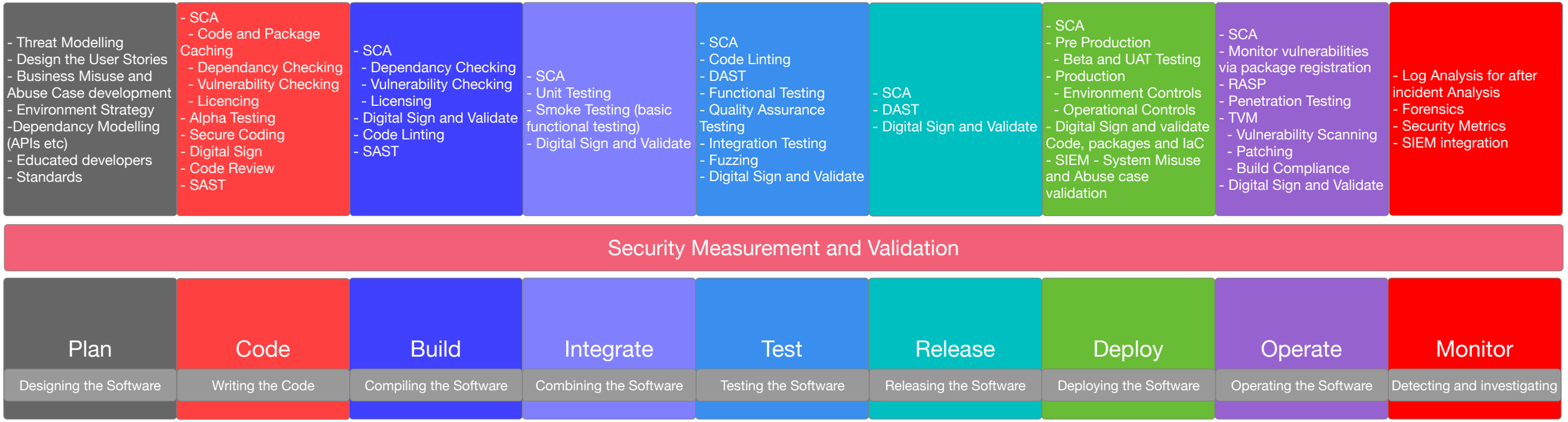
Plan	Code	Build	Integrate	Test	Release	Deploy	Operate	Monitor
- What are the threats - Who are the users - What regulations do we need to follow? What is the long term strategy? - How do we manage 3rd party APIs - Whats standards are applicable? Developers trained?	- Trust the source of the code? - 3rd party packages/ scripts/source code/test code vulnerable? - Licenced code in use? - Poor coding or error? - Malicious code present? - Coding standards being followed? - Creds/Secrets securely implemented	- Vulnerable code or packages present? - Are dependant packages still safe? - Licensed to use the dependant packages? - Trust the source? - Are secrets safe?	- Vulnerable packages present? - Dependancies met? - Is everything talking as expected? - Can we trust the components? - If 3rd parties are developing part of the application can we trust they have done things securely?	- Are we using vulnerable code/components? - Can we break the logic? - Have we tested all misuse and abuse cases?	- Are we using vulnerable code/components? - Can we trust the components? - Are we licensed to use the dependant packages?	- Are we using vulnerable code/components? - Are deployed components licenced? - Blast Radius - Exposed credentials? - Down revisions of code in use? - Is “on the fly” alteration of code outside of the pipeline possible?	- Is everything available? - Is the service running ? - Are there any vulnerabilities? - Is anybody attacking the infrastructure?	- Can we determine what happened in a breach - Are we maintaining the environment effectively - Do we have any vulnerabilities in either the infrastructure, application or both? - Do we obtain forensics after a breach? - Are we complying with regulation?
Designing the Software	Writing the Code	Compiling the Software	Combining the Software	Testing the Software	Releasing the Software	Deploying the Software	Operating the Software	Detecting and investigating

Security Measurement and Validation

What Tools do we have in the kit bag?



What are the Controls?



In Summary

I have tried to decompose and educate the uninitiated to the world of modern application security. Why, because it is so very different from traditional application delivery mechanisms and a world away from normal infrastructure security. The principles remain the same but the approach to securing is different.

As security professionals we need to understand, adapt and evolve to meet the challenges presented by modern software development practices. I hope this has helped you on that Journey.

Quote : ""It is not the strongest of the species that survive, nor the most intelligent, but the one most responsive to change."
– Charles Darwin

Questions?

Contact: Rob Campbell
Email : cosac@assuredcontrol.com
Linkedin : robert-campbell-security

These slides and other goodies – https://www.assuredcontrol.com

I am willing to provide the layered source diagrams for you to modify and use. They are in Omnigraffle format but I can convert to Visio. Just email and ask.

Appendix A – Pipeline Components

Pipeline Components	Description
Integrated Development Environment	Provides the developers the tools to; - Security and other Plugins - Compiler - Interpreter - Source Code Editor. - Debugger - Build Automation. - Versioning - Class and Object Browsers
Repository	Supports the Software Development Process by enabling collaboration via sharing and reuse. - Acts as a storage location for code and packages - Package Manager - Manages malware and vulnerabilities within the repo - Uses strong identity, Permisions and Digital signatures to help control threat of malware and build trust.
Repository Manager	A Repository Manager deals with; - User triggering the build (whether manually or by committing to revision control) - Modules built - Sources used (commit id, revision, branch) - Dependencies used - Environment variables - Packages installed
Package Manager	A package manager deals with - Package distributions of software and data in archive files. - Packages metadata, such as - Software's name, - Description of its purpose, - Version number, - Vendor, - Checksum or digital signature, - Vulnerability information - A list of dependencies necessary for the software to run properly.

Appendix A – Pipeline Components

Pipeline Components	Description
Build Tool	Automates the creation of executable applications from source code. - compiling - linking - packaging
Container/Hypervisor Engine	-Provides the infrastructure to support the running of a virtualised environment to run applications.
Container/VM Registry	A repository, or collection of repositories, used to store container/VM images for application development and IaC type deployments.
Container/VM Orchestration	Manages the infrastructure used to support the running of virtualised environments. - Provisioning and deployment of containers - Redundancy and availability of containers - Scaling up or removing containers to spread the application load evenly - Movement of Containers between hosts - Allocation of resources to and between containers - Exposing Container Services - Health Monitoring - Load balancing - Application configuration - Key and Certificate Management

Appendix B –Security Tools

Security Tooling	What it does	Weaknesses	Core Capability
Software Composition Analysis (SCA)	- Discovers OS dependancies - Versioning - Usage Info - Alerts on risks and policy violations - Licensing - Policy based blocking with workflow	- Vulnerability detection only as good as the intelligence. - Quality of data on vulnerabilities can be variable.	Detect Prioritise Prevention
Static Application Security Testing (SAST) AKA - White box Testing	- Finds Coding Errors - Analyses the code and binary without executing - Doesn't need to be deployed - Finds vulnerabilities early so fix is quicker and cheaper. - Runs in IDE or Repo	- Won't discover run-time vulnerabilities - Lots of False Positives and False Negatives	Detect Remediate
Dynamic Application Security Testing (DAST) AKA - Black Box Testing	- Analyses the running application - Finds Runtime errors	- Won't find Coding errors	Detect Remediate

Appendix B – Security Tools

Security Tooling	What it does	Weaknesses	Core Capability
Runtime Application Security Protection (RASP)	- Works within the application whilst running - Controls Application Execution - Looks for unusual behaviour - Can terminate the a suspicious session	- Can be performance heavy	Detect Prevention
Digital Signatures	- Provide integrity throughout the pipeline. - Enables Trust	- Requires PKI and all associated controls to protect the infrastructure.	Digital Trust Validation
Penetration Testing	- Includes human insight -Tests infrastructure as well as application	- Point in time - Dependant on Testers skill and experience - Massive variation in technology means you need to pick and choose your testing partner/testers.	Detect Prioritise