IMPLEMENT DISASTER RECOVERY

Objectives

- Provide an overview of disaster recovery (DR) response and personnel in the disaster recovery plan (DRP)
- Compare communications methods
- Describe DRP assessment and restoration
- Outline disaster recovery plan tests, lessons learned, and communicating test results
- Describe training and awareness for disaster recovery planning

DISASTER RECOVERY RESPONSE

- DR planning involves ensuring that managers can help the organization recover to an acceptable level from any type of catastrophic event
- A cataclysmic event is relative to the organization as it can be a single drive ransomware attack to an entire facility or campus being put out of action
- The DRP should contain detailed steps for recovering from any kind of data loss or physical disaster

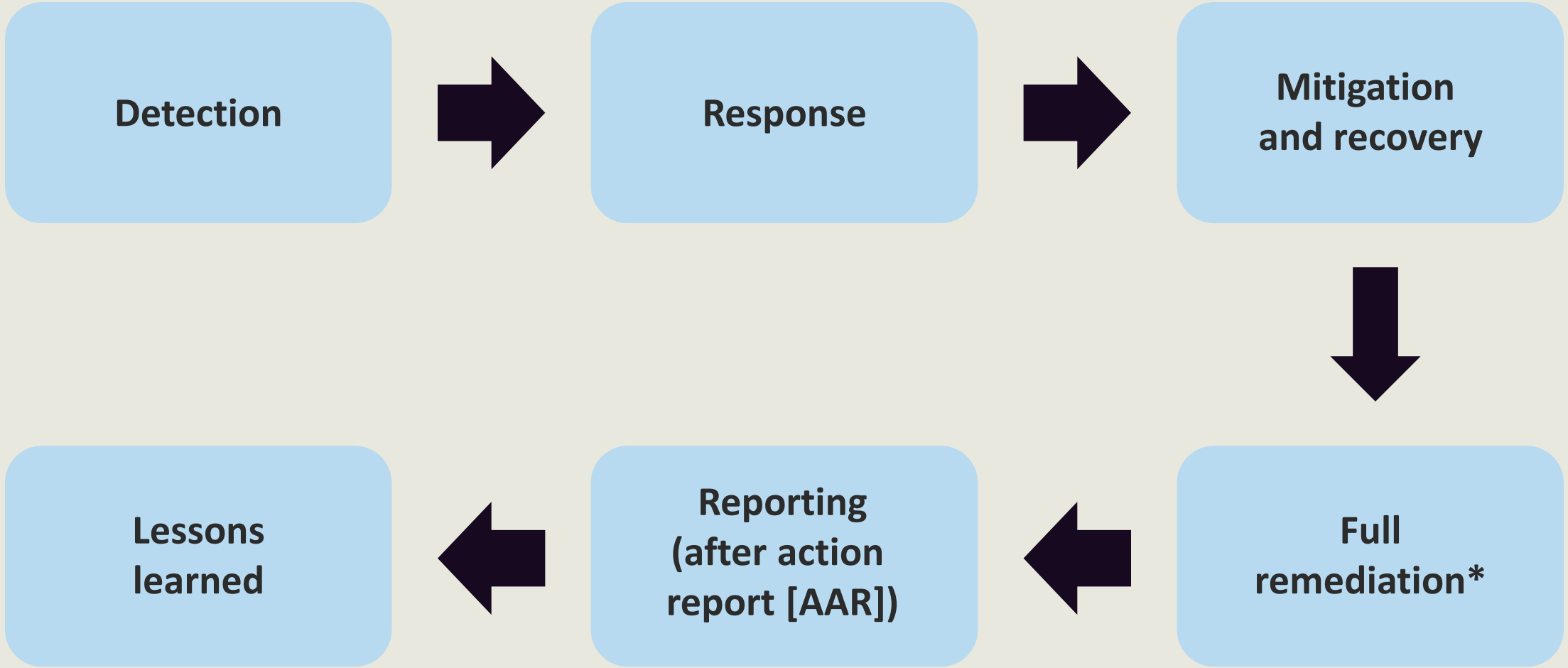
DISASTER RECOVERY PLANNING

- Step-by-step instructions on how to recover each aspect of critical systems, applications, and data
- Order of succession and command
- Backup and restore plans with order of restoration
- Contact information for key stakeholders

DISASTER RECOVERY PLANNING

- Contact information for law enforcement, legal, insurance, media outlets, and more
- Location of hot spares, software and CD keys, security access keys and failsafe passwords, and other valuable documentation
- DRP site locations and descriptions
- **These plans should also be physical hard copies**

DISASTER RECOVERY PLANNING LIFE CYCLE



PERSONNEL IN THE DR PROCESS

Organizational charts

Have physical and digital copies of the roles and responsibilities

Lines of succession

Know who will step up in case an individual is incapacitated

Contact information

Have comprehensive contact information and emergency contacts

PERSONNEL IN THE DR PROCESS

- Most large organizations have a separate department or third-party vendor that supplements HR and legal to offer emergency management, counseling, and personal duress assistance
- Arrangements should be made to deal with post-traumatic stress and other mental health challenges that arise after a catastrophic event
- Disaster recovery must be addressed in security training and awareness programs
- Accommodations should be made to temporarily or permanently replace key roles and responsibilities in contingency planning

DRP PERSONNEL WITH RACI CHARTS

R – Responsible **A** – Accountable **C** – Consulted **I** – Informed

	GRC* department	Legal department	Security team	IT operations
Establish the continuity requirements	R/A	C	C	I
Build the governance scheme	R/A	C	C	I
Assess recovery sites	A	I	R	R
Build the plan	I	I	A/R	R
Conduct tests and exercises	I	I	C	A/R

**GRC – Governance, Risk, and Compliance*

COMMUNICATION METHODS

- A disaster recovery communication plan conveys the necessary tasks for sharing information with employees, contractors, customers, and other core stakeholders during and after a catastrophic event
- This information must be stored redundantly and in various formats – not just digitized on a single disk or database
- **Backup communication plans include different ways to communicate in an emergency other than the corporate VoIP solutions**

DISASTER RECOVERY COMMUNICATION MODES

- Telephone land lines
- Cellular phones
- Two-way radios
- Satellite phones
- Citizen band radios (CB)
- Short wave radios

EMERGENCY COMMUNICATION PLANNING

01 Preserve current contact information data backups

02 Assemble a disaster communication team

03 Collect updated emergency contact information

04 Inform all internal and external stakeholders

05 Determine the communication strategies

06 Practice and rehearse various scenarios

ASSESSING THE DISASTER RECOVERY PLANS

- Assessment of the disaster recovery plan should begin with making sure that the **executive management or steering committee has approved the plan** and all elements
- Feedback and changes should be implemented in an iterative manner
- A solid plan will have senior management review and sign off on a regular basis
- This ensures that a particular role is ultimately responsible for due diligence and due care of the DRP

ASSESSING THE DISASTER RECOVERY PLANS

- The DRP should **follow a standardized, proven, and logical format**
- There must be a concise and well-organized approach to the recovery plans
- The table of contents should follow a logical path for the plan, assembling the teams, categorizing events, and executing recovery steps, etc.
- This document should be one of the most well constructed plans that the organization has based on the value to continuity of operations

ASSESSING THE DISASTER RECOVERY PLANS

- **The recovery team leaders should be evaluated and retrained on a regular basis**
- Identify the individual that supervises the recovery team leaders:
  - CIO, CISO, CEO, etc.
- Every element of the recovery plan must have a proven team leader and members who are capable to develop, maintain, and execute the plan in an emergency

ASSESSING THE DISASTER RECOVERY PLANS

- Recovery teams must perform regular gap analysis to **assure that any new or changed critical business processes are included in the plan**
- To be effective, managers must know exactly what processes need to be recovered, in the right timeframes, and all dependencies
- For example, recent challenges with the supply chains will force security professionals to reassess metrics such as mean time to respond (MTTR) and its effect on the DRP

ASSESSING THE DISASTER RECOVERY PLANS

- **The DRP should be as "event neutral" as possible**
- Since planners cannot develop a detailed scheme for every possible scenario, there should be a macro approach when feasible
- Scenarios such as loss of personnel, communication, data, and facilities should be addressed for a wide variety of negative events
- **Ultimately the DRP assessment will come from the lessons learned aspect of AAR**

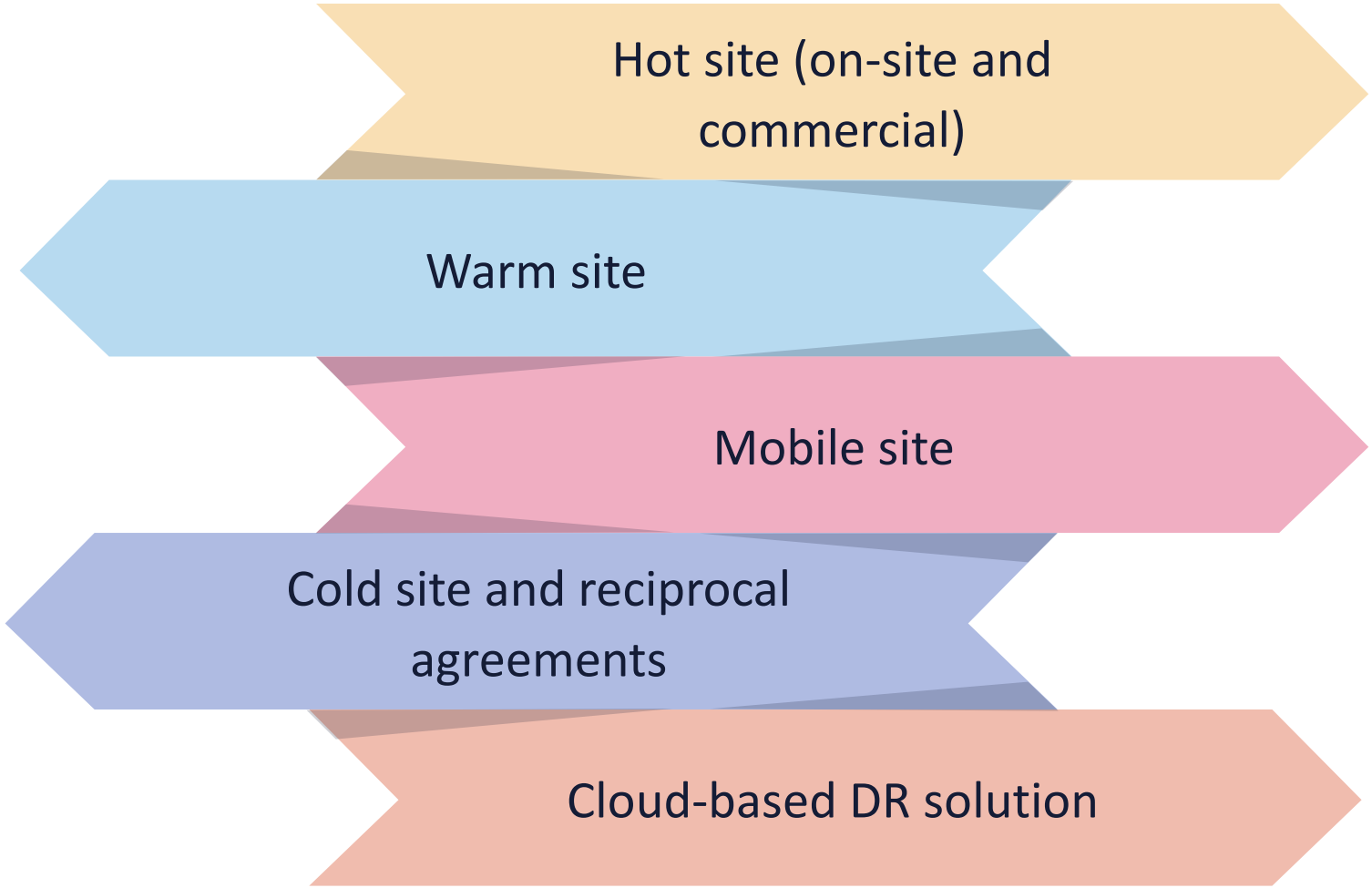
RESTORATION FROM DISASTERS

- Recovering from a disaster is often a long and even gradual process
- An organization may be in a "perpetual" state of disaster recovery (i.e., post-pandemic remote employees, supply chain disruptions)
- Managers should first take care of personnel health and safety:
  - Tend to physical and mental well-being on a regular, ongoing basis

RESTORATION FROM DISASTERS

- Follow through on filed insurance claims and apply for available disaster assistance, such as from the Federal Emergency Management Agency (FEMA)
- Return from DRP site in an organized and systematic manner
- Document all actions and add them to the lessons learned database

RESTORATION SITE SOLUTIONS



READ-THROUGH (PLAN REVIEW)

- The read-through (plan review) is where the business continuity plan (BCP) owner and business continuity team discuss the BCP
- This is a type of checklist test that is useful for training new members of a team, including the business function owner
- It is constructed from brainstorming, surveys, and questionnaires
- The goal is to create a baseline and then find gaps and inconsistencies within the plan or with the organization

TABLETOP TESTING

- Participants gather in a room to execute documented plan activities in a stress-free environment using visualization techniques
- Teams will use blueprints, topological diagrams, or computer modeling and graphics to effectively demonstrate whether team members know their duties in an emergency and if they need training
- This will find additional documentation errors, missing data, and inconsistencies across disaster and business continuity plans

WALKTHROUGH TESTING

- Is a planned but limited rehearsal of a possible incident designed to evaluate an organization's capability to manage that incident
- Provides opportunities to build and improve on the read-throughs and tabletop tests
- Should improve the organization's future responses and enhance relevant competences of participants
- Is often done by scheduling drills for each department or building separately without affecting operations

SIMULATION TESTING

- This more elaborate test/drill will utilize established business continuity resources, recovery sites, backup equipment, services from recovery vendors, communication, and transportation recovery modes
- It will determine if business continuity management procedures and resources function in a realistic scenario
- This may be the most elaborate test most commercial entities ever conduct
- It can require sending teams to alternate sites to restart technology as well as business functions

PARALLEL TESTING

- This elaborate exercise involves bringing the recovery site to a state of operational readiness, but maintaining a low level of operations at the primary site
- The staff are relocated, backup tapes are transferred, and operational readiness established in accordance with the DRP at the recovery site
- Operations at the primary site continue with a skeleton crew
- This will have a substantial impact on operations and may demand crisis actors and other external consultants

FULL-INTERRUPTION TESTING

- Operations are completely shut down at the primary site to fully emulate the disaster
- The value proposition of the organization ceases to be delivered
- The enterprise transfers to the recovery site in accordance with the disaster recovery plan (hot, warm, cold, cloud)
- This is a very thorough test, which is also expensive and typically cost-prohibitive
- It has the capacity to cause a major disruption of operations if the test fails

LESSONS LEARNED

- Lessons learned is the knowledge and wisdom gained from the process of conducting the test/drill, program, project, or exercise
- This information will also be included in the AAR post-disaster
- Formal sessions are usually held at the project close-out or near the completion of the initiative

LESSONS LEARNED

- Lessons learned should be recognized and documented at any point during the life cycle to
  - Share and use knowledge derived from an experience
  - Endorse the recurrence of positive outcomes
  - Prevent the recurrence of negative outcomes
- Although someone may be held ultimately responsible, the goal of AAR should NOT be blamestorming or scapegoating

COMMUNICATING TEST RESULTS

- Once the after-action reporting and lessons learned are documented, the results must be adequately communicated to a variety of internal and external stakeholders:
  - Team members
  - Boards, committees, and executives
  - Insurers
  - Regulators
  - Press and media
  - Public declarations (website)
  - Existing and potential strategic partners
  - Existing and potential customers/clients

COMMUNICATING TEST RESULTS

- Align the AAR reporting with security guidance, policies, standards, and guidelines
- Maintain a consistent reporting structure over the life cycle for improved results
- Purge outdated and obsolete communications and data from subsequent reporting
- Engage the report consumers and stakeholders regularly to get feedback for possible improvements

DRP TRAINING AND AWARENESS BEST PRACTICES

- Remember that there is no "one-size-fits-all" DRP training solution
- Define the precise training objectives at the outset
- Ensure that all internal stakeholders are included in the awareness
- Consider the importance of rotation of duties and dual operator principles with disaster recovery
- Prioritize and construct specific risk scenarios for exercises and training

DRP TRAINING AND AWARENESS BEST PRACTICES

- Run joint training and awareness seminars with other groups such as Security Operation Center (SOC), incident response teams, security guards, and facilities management
- Vary the modalities of training between computer-based, webinars, classroom tabletop sessions, workshops, live drills, and exercises
- Consider hiring crisis actors when more elaborate training and exercises is cost-prohibitive

SECURING THE SDLC AND SOFTWARE DEVELOPMENT ECOSYSTEM

Objectives

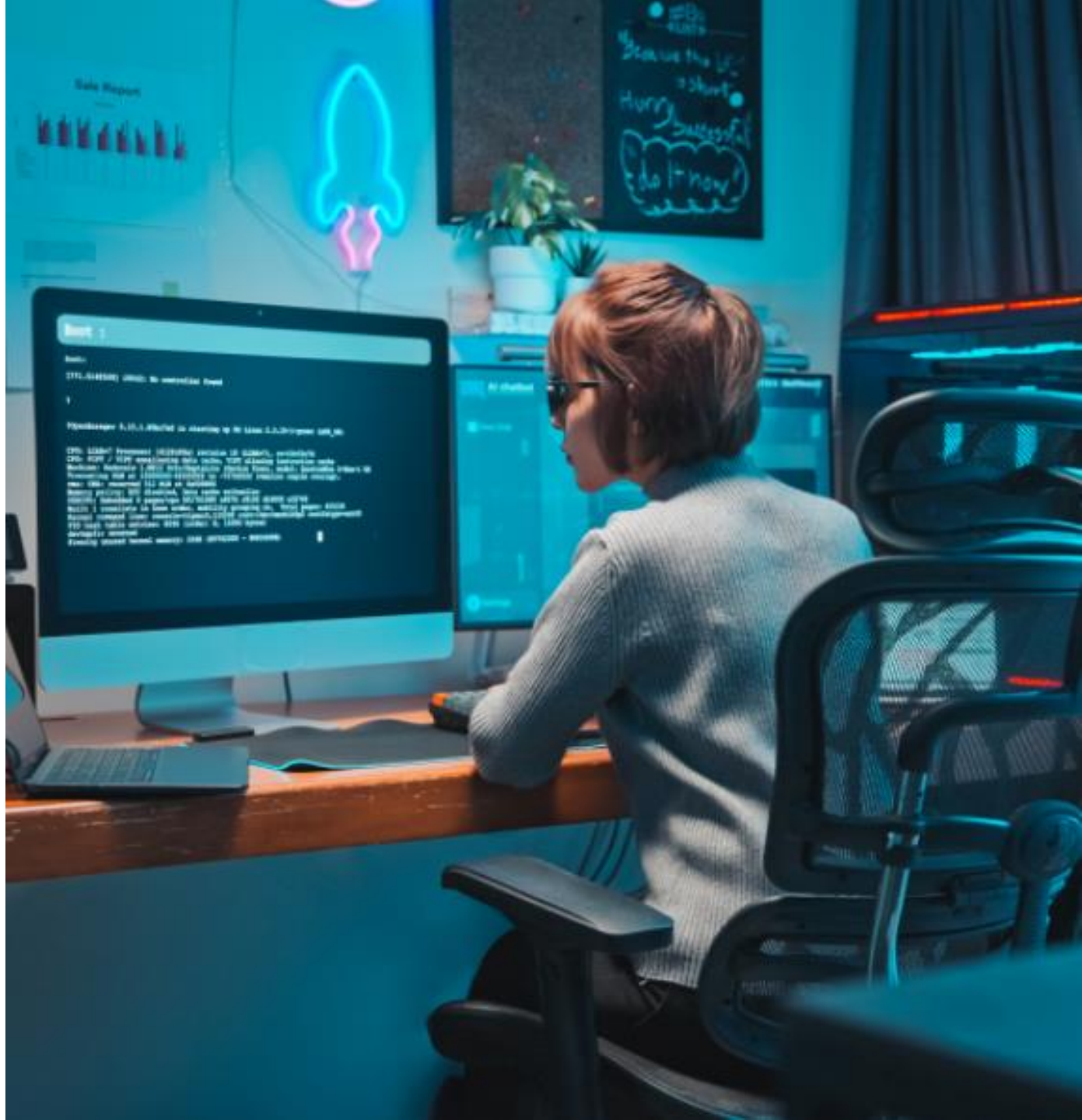
- Describe development methodologies and maturity models
- Provide an overview of DevOps operation, maintenance and change management, and integrated product teams
- Compare programming languages, libraries, integrated development environments, toolsets, and runtime environments
- Describe continuous integration (CI)/ continuous delivery (CD) and Agile life cycles
- Provide an overview of software configuration management (SCM), code repositories, and application security testing methods

SECURE BY DESIGN

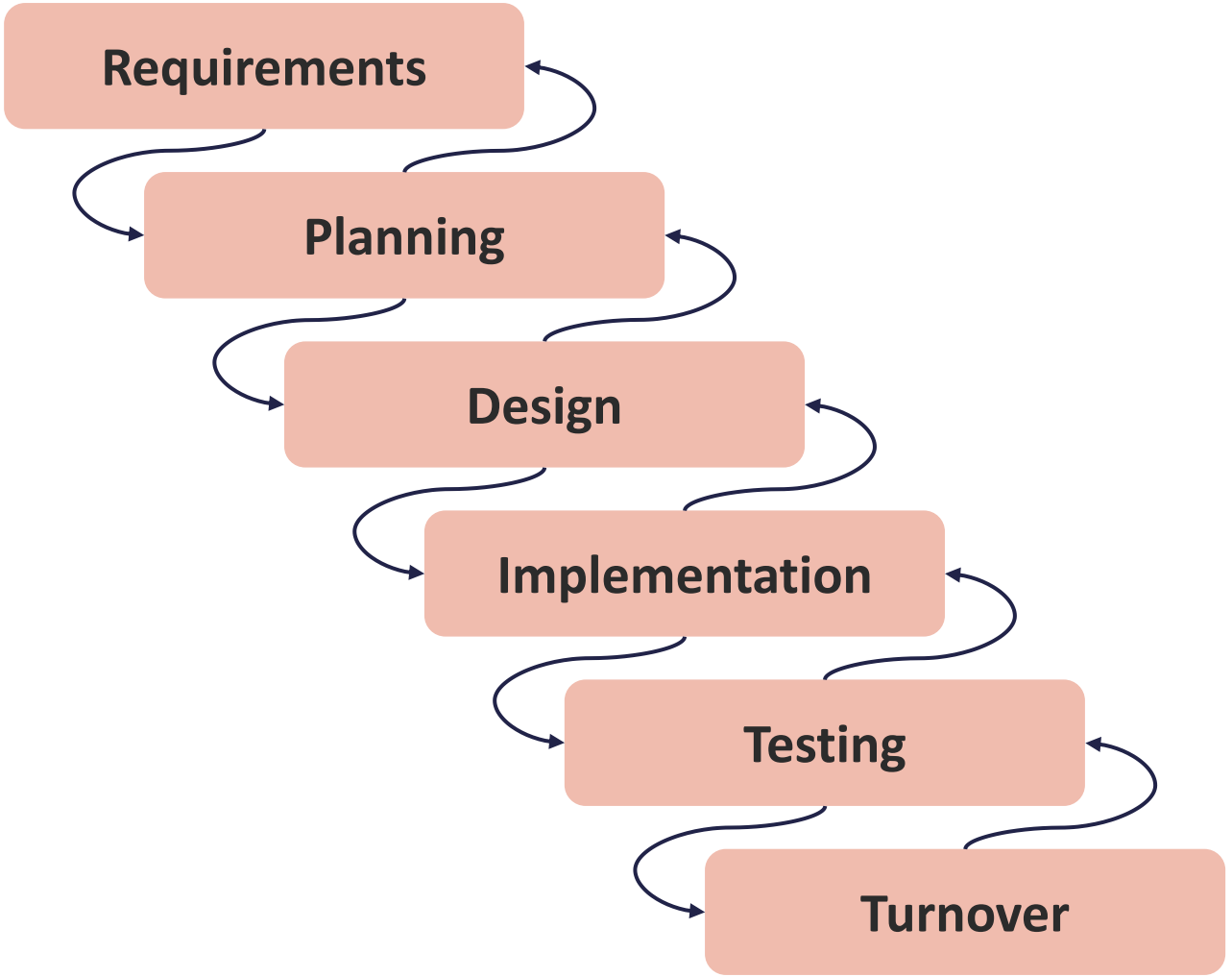
- The custom or outsourced program or application is developed with security integrated into the entire software development life cycle (SDLC)
- Security by design ensures apps are constructed with security in mind from the outset using the following principles:
  - Least privilege - limits access to only what is necessary
  - Segregation of duties - partitions responsibilities to prevent abuse of the application
  - Defense in depth - uses several layers of security
  - Failing securely - handles errors carefully
  - Open design - avoids security through obscurity

SECURE BY DEFAULT

- This application design consideration assumes the application is natively secure without any modifications or additional controls:
  - Example: a server application has certain possible unsecure functions, but they are disabled by default at deployment based on Infrastructure as Code (IaC)
  - Example: building an application solution on tested and proven containers and microservices linked by secure application programming interfaces (APIs)



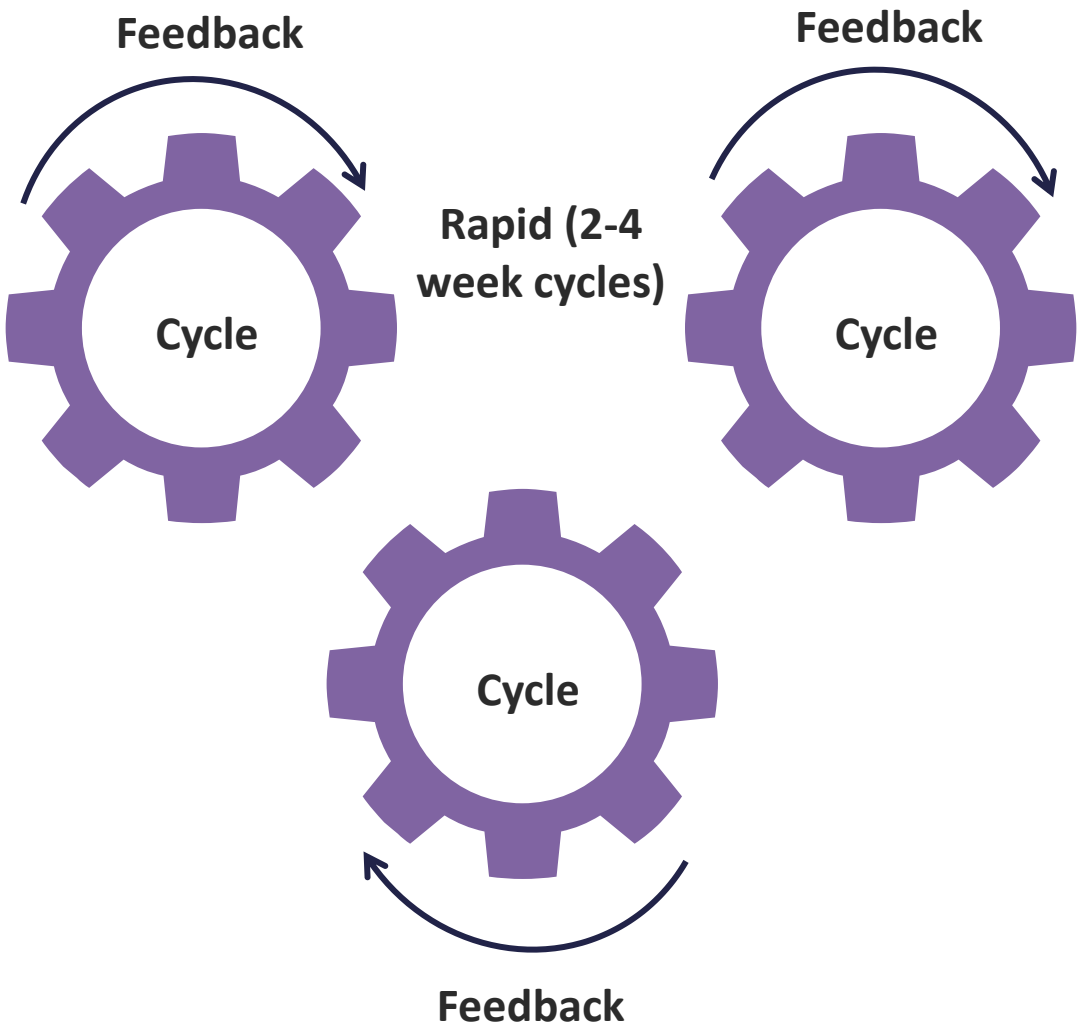
DEVELOPMENT METHODOLOGIES: WATERFALL



DEVELOPMENT METHODOLOGIES AGILE

- Agile is popular and excellent for smaller and rapidly developed projects
- It is an evolutionary approach that is measured in weeks and involves collaboration between cross-functional teams
- Agile is flexible, adaptable, not predictable, and testing is done during development
- Estimates (i.e., budget, schedule, etc.) get more realistic as work progresses, because important issues are discovered earlier

DEVELOPMENT METHODOLOGIES: AGILE

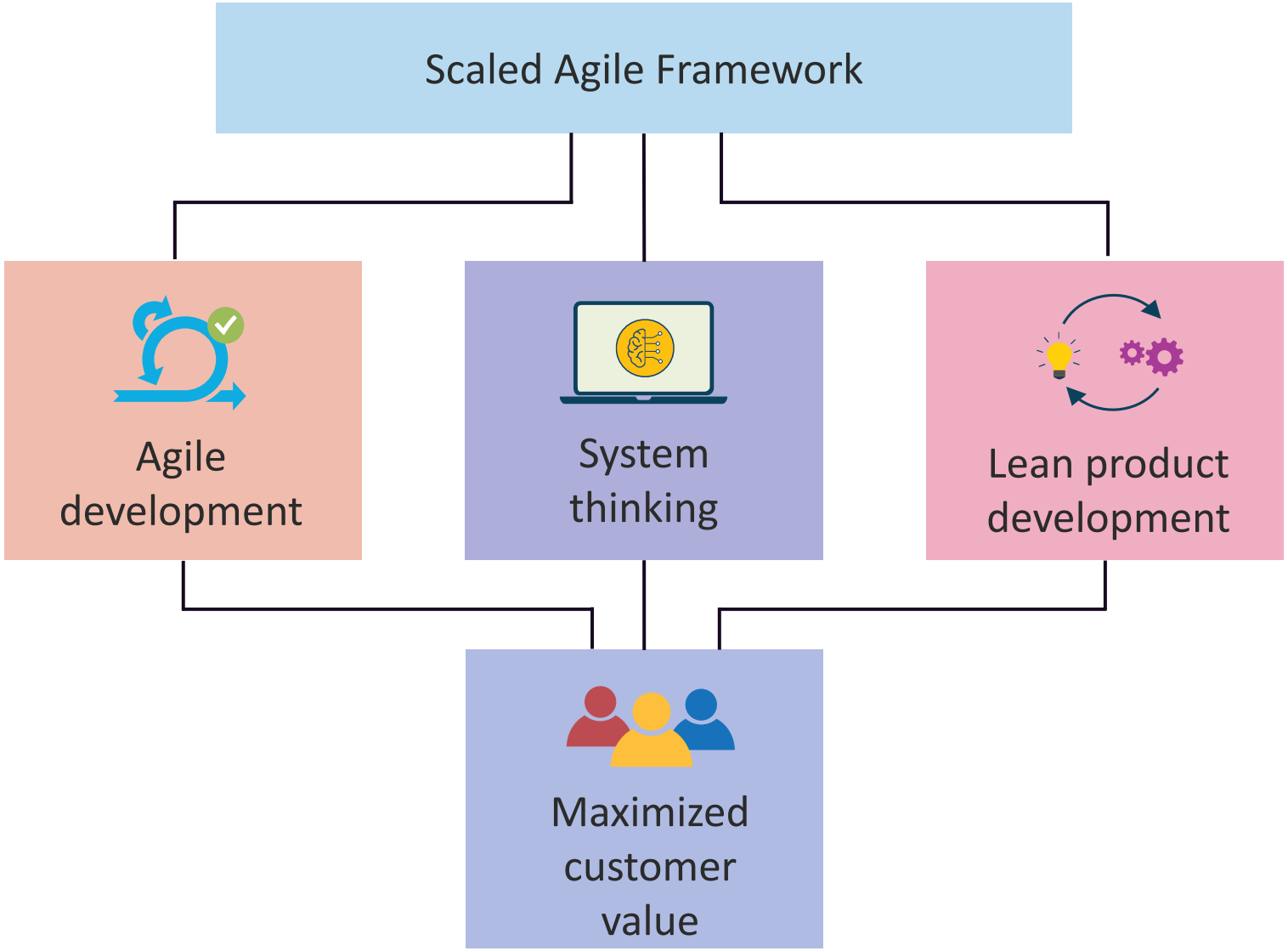


SCALED AGILE FRAMEWORK

- The Scaled Agile Framework (SAFe) is a collection of structural and workflow outlines for deploying Agile practices at scale
- It is a free and open online knowledge base that offers organized direction on roles and responsibilities
- The framework shows how to plan and manage the workflow
- The goal is to meet the challenges of scaling Agile development in large, complex, distributed, or compliant heavy environments

SCALED AGILE FRAMEWORK

- There are four pillars of SAFe:
  - **Alignment** assures that all participants are working on the same goals
  - **Built-in quality** focuses on quality from the outset
  - **Transparency** makes information observable and available
  - **Program execution** delivers value through effective implementation


Scaled Agile Framework
Agile development
System thinking
Lean product development
Maximized customer value

DevSecOps

- DevSecOps is a framework that integrates security into all phases of the SDLC
- It involves introducing security earlier in the software development process and expanding the collaboration between development, operations, and security teams
- DevSecOps aims to reduce the risk of releasing code with security vulnerabilities and to automate the protection of the software delivery cycle

CAPABILITY MATURITY MODEL (CMM)

- The Capability Maturity Model is a methodology and initiative used to advance and enhance an organization's system and software development processes
- The model consists of a five-level evolutionary path of progressively prepared and systematically more mature processes
- It is like ISO 9001 standards that specify an effective quality system for manufacturing and service industries

CAPABILITY MATURITY MODEL

Initial (chaotic)
This level is chaotic, ad hoc, and prone to individual heroics
Level 1

Repeatable (implicit)
Process is not codified or defined and is still vulnerable to inconsistency
Level 2

Defined (early explicit)
Process is defined and documented as a standard business process
Level 3

Managed (mature explicit)
Process is controlled and can be adjusted and adapted to particular projects without measurable losses of quality
Level 4

Optimized (purely explicit)
Process management includes deliberate process optimization and improvement
Level 5

CMM LEVEL 1

- At the **initial (chaotic) level 1**, processes are disorganized, even chaotic
- Success most often depends on individual heroics and is not considered to be repeatable
- Processes are not sufficiently defined or documented for them to be replicated
- Decision-making is a free-for-all and based on intuition and existing experience
- Decision-making authorities are poorly defined, with no key risk indicators (KRIs), key performance indicators (KPIs), critical success factors (CSFs), or real meaningful metrics
- Results are inconsistent and often unaligned with any executive leadership

CMM LEVEL 2

- At the **repeatable (implicit) level 2**, fundamental project management is established, and successes could be repeated, as the obligatory processes have been established, somewhat defined, and documented
- Decision-making is based on rote adherence to poorly-aligned standards and practices, and data provided to decision-makers is often superficial, not codified, and cannot hold up to scrutiny
- Roles and responsibilities are unclear, and it is common for people to make decisions without authorization
- Risk is defined purely qualitatively and without much support of expert judgment or historical precedent
- Established KRIs and metrics, if any, are questionable as risk terminology, taxonomy, and policy is superficial or non-existent

CMM LEVEL 3

- At the **defined (early explicit) level 3**, the enterprise has developed its own standard software process through greater attention to documentation, standardization, and integration
- Standardized terminology exists and assessments are more up-to-date and better supported
- Visibility is improved as more robust and defensible analysis exists
- Well established standards, security teams, and steering committees exist
- Components like service desk and ITIL 4 practices are put in use
- Risk registers, KPIs, and KRIs are defined
- Meaningful metrics and calibrated and more precise semi-quant and quants are evident

CMM LEVEL 4

- At the **managed (mature explicit) level 4**, the processes are controlled and can quickly be adjusted and adapted for development projects, security initiatives, or other endeavors without measurable loss of quality
- Quality is visible and risk registers and assessments are up-to-date
- Data is actively used, and risk treatment/handling is adapted accurately based on more quantitative analysis
- Indicators and metrics are well-defined and tested
- Methods for assurance and certification are established if pertinent
- This is the highest level that most DevSecOps initiatives and organizations can hope to meet

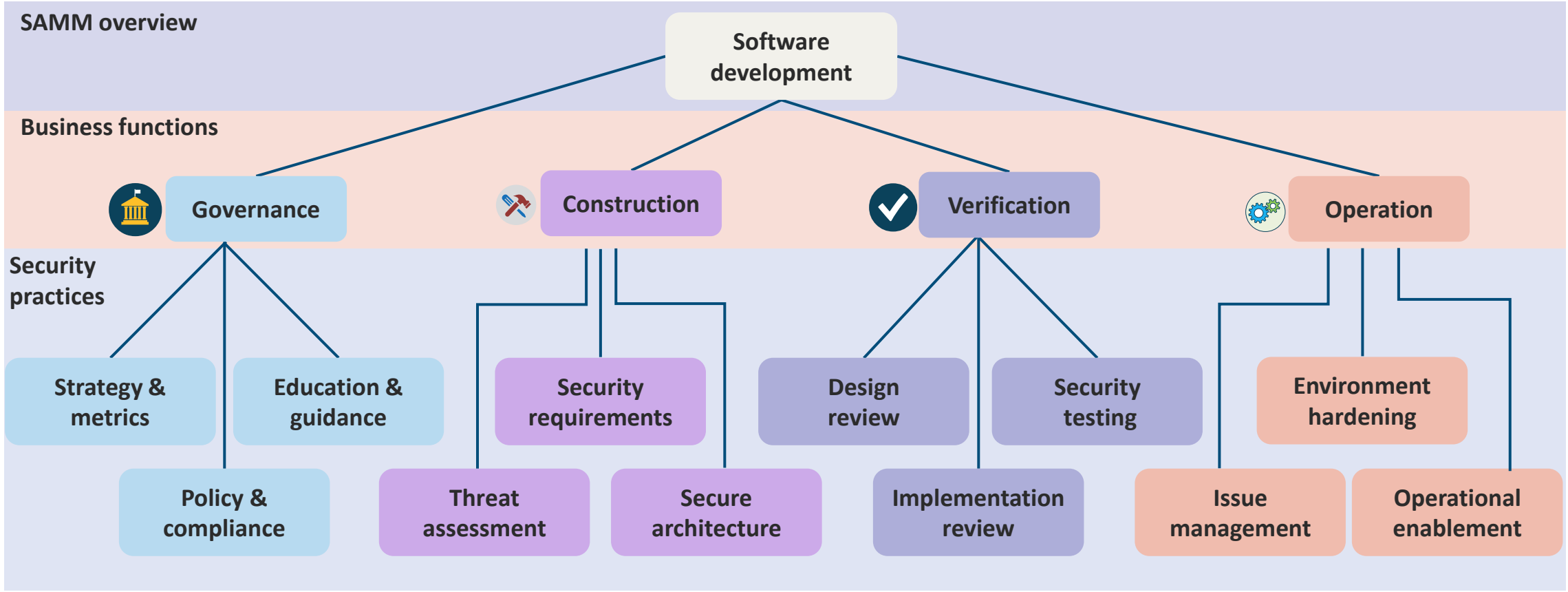
CMM LEVEL 5

- At the **optimized (purely explicit) level 5**, the process management has attained everything at level 4, including deliberate process optimization and continual improvement
- Level 5 is rarely achieved, but is still possible with proper leadership and resources
- For example, ITIL 4 mastery and maximum software development proficiency would be demonstrated in this organization

SOFTWARE ASSURANCE MATURITY MODEL (SAMM)

- The Software Assurance Maturity Model is an open framework from OWASP to assist organizations in developing and deploying a secure software delivery strategy that is focused on the detailed risks facing the enterprise
- The resources offered by SAMM will assist in
  - Appraising the organization's current software security initiatives
  - Constructing a well-adjusted software security assurance program using established iterative processes
  - Establishing tangible continual improvement methodologies to a software security assurance program
  - Defining and gauging security-related tasks throughout the enterprise

SAMM FRAMEWORK



"Software Assurance Maturity Model." OWASP.org. Accessed June 8, 2021. https://owasp.org/www-pdf-archive/SAMM_Core_V1-5_FINAL.pdf.

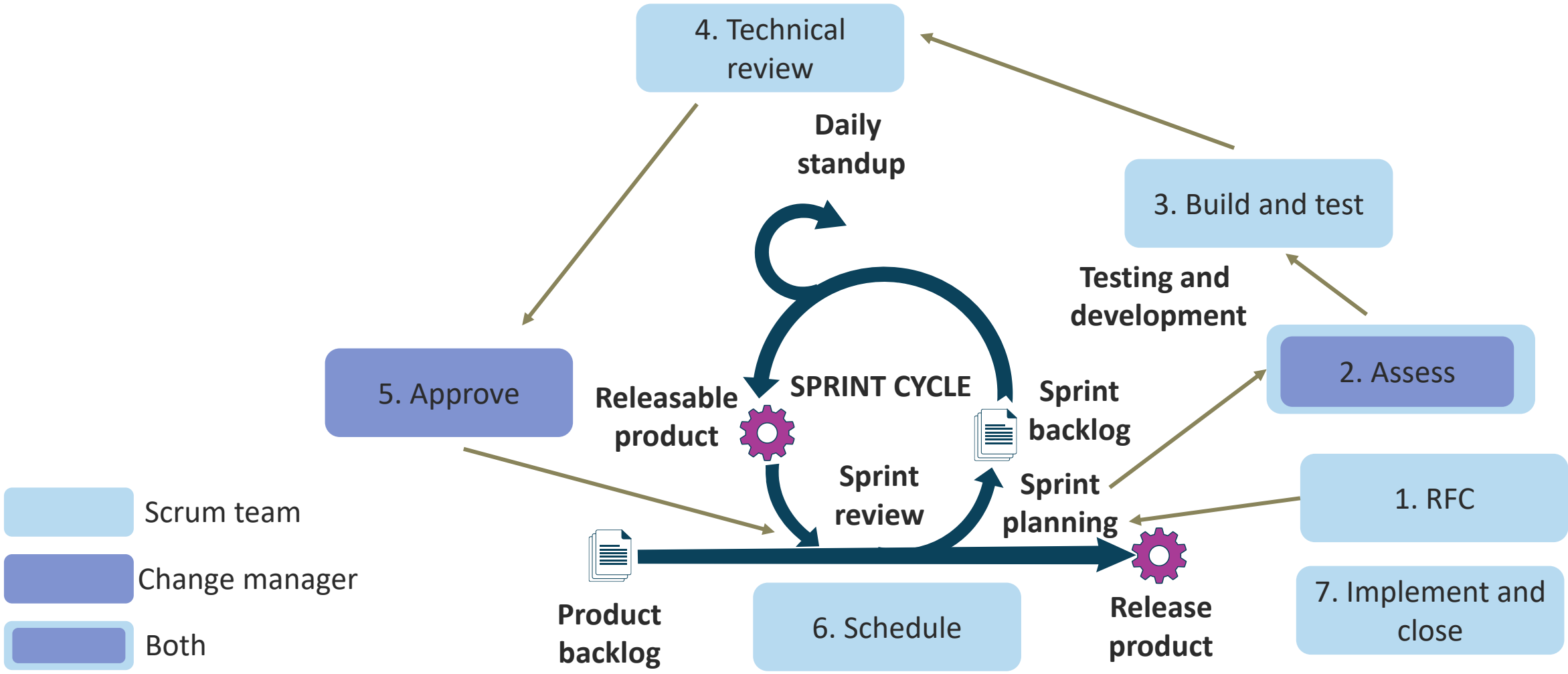
DevOps OPERATIONS

- DevOps operations are the tasks and processes that empower application development teams to successfully plan, construct, distribute, and workflow software products, applications, and systems
- DevOps operations include processes like version control, Agile and CI/CD development, Infrastructure as Code, configuration management, and constant monitoring and visibility
- Operations target improved collaboration, quality, speed, and dependability of software deployment

DevOps MAINTENANCE

- DevOps teams strive to certify application reliability, high availability, and performance
- Maintenance also involves reinforcing security and governance
- To maintain DevOps, developers will
  - Evaluate and improve code frequently
  - Plan for failure in a proactive way
  - Gain visibility into software performance
  - Fix bugs and errors
  - Track user feedback and experience (CX/UX) and address them in near real-time
  - Confirm that programmers are accountable for assuring that code works in production

DevOps Change Management (ITIL 4)



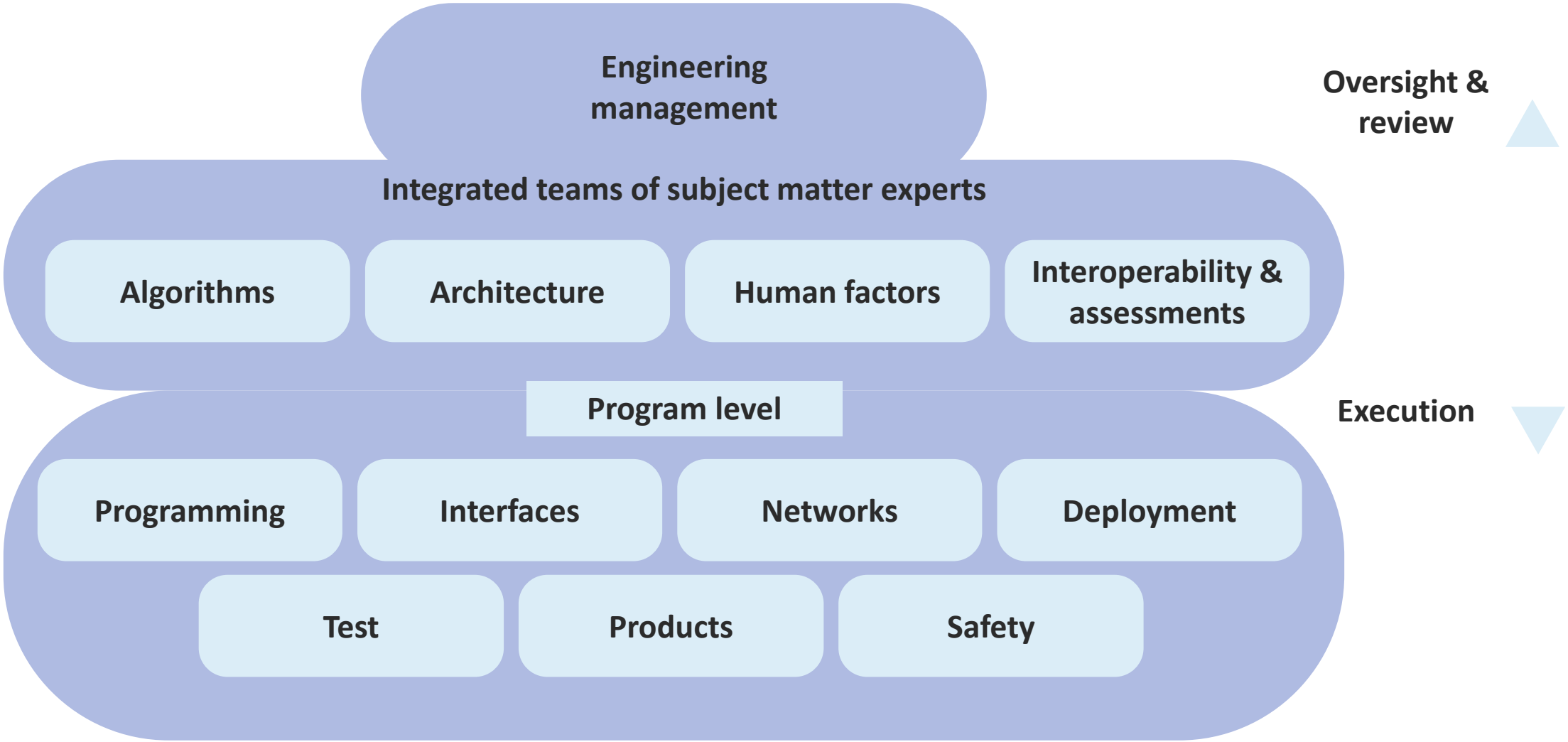
INTEGRATED PRODUCT TEAMS (IPTs)

- An IPT is defined as a multidisciplinary team of people who are collectively responsible for delivering a well-defined product or service solution
- The Department of Defense (DoD) has adopted IPTs as their preferred approach for systems and software acquisition
- IPTs are maintained by different subject matter experts and can be leveraged throughout the entire development life cycle

INTEGRATED PRODUCT TEAMS (IPTs)

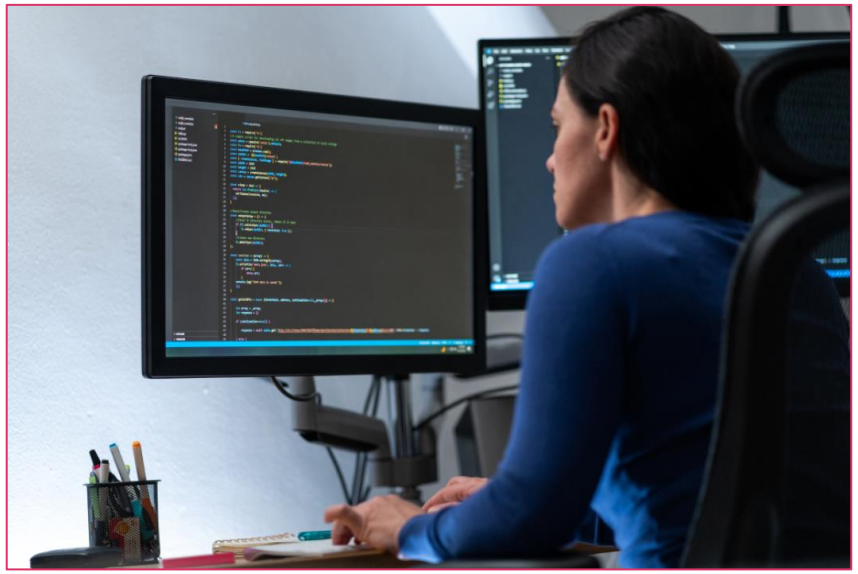
- These groups of key individuals represent varying ranges of expertise with the common goal of delivering the best product or service
- This cross-functional expert judgment also supports product acquisition activities and the development of system and software solutions

IPT FRAMEWORK

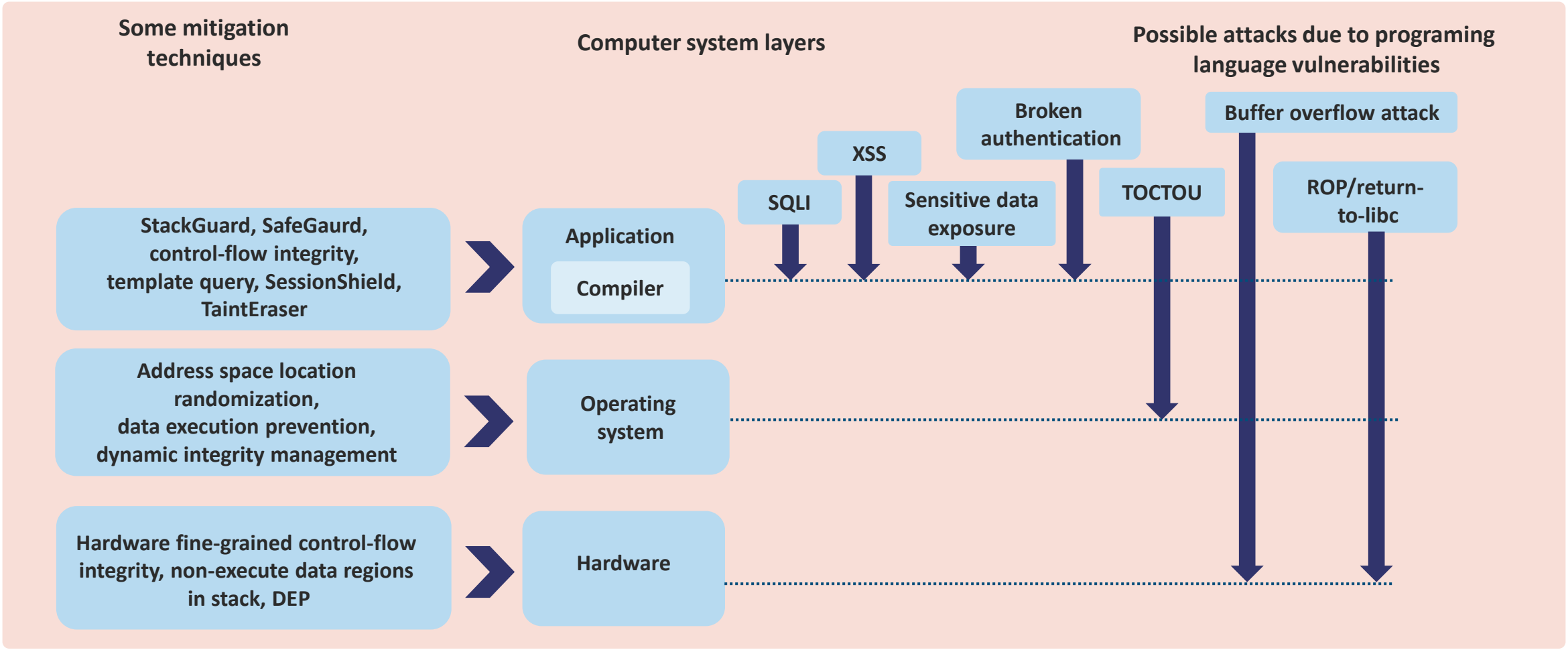


SECURING LANGUAGES, LIBRARIES, AND RUNTIME ENVIRONMENTS

- A critical aspect of DevSecOps and security by design is to secure and harden the programming language, libraries, and the eventual runtime environment
- Most developers and development teams will rely on tools and engines in the integrated platforms to provide a variety of tests, checks, and validation of code
- Teams will also leverage regular peer reviews and automated visibility processes



COMMON PROGRAMMING SECURITY CONTROLS

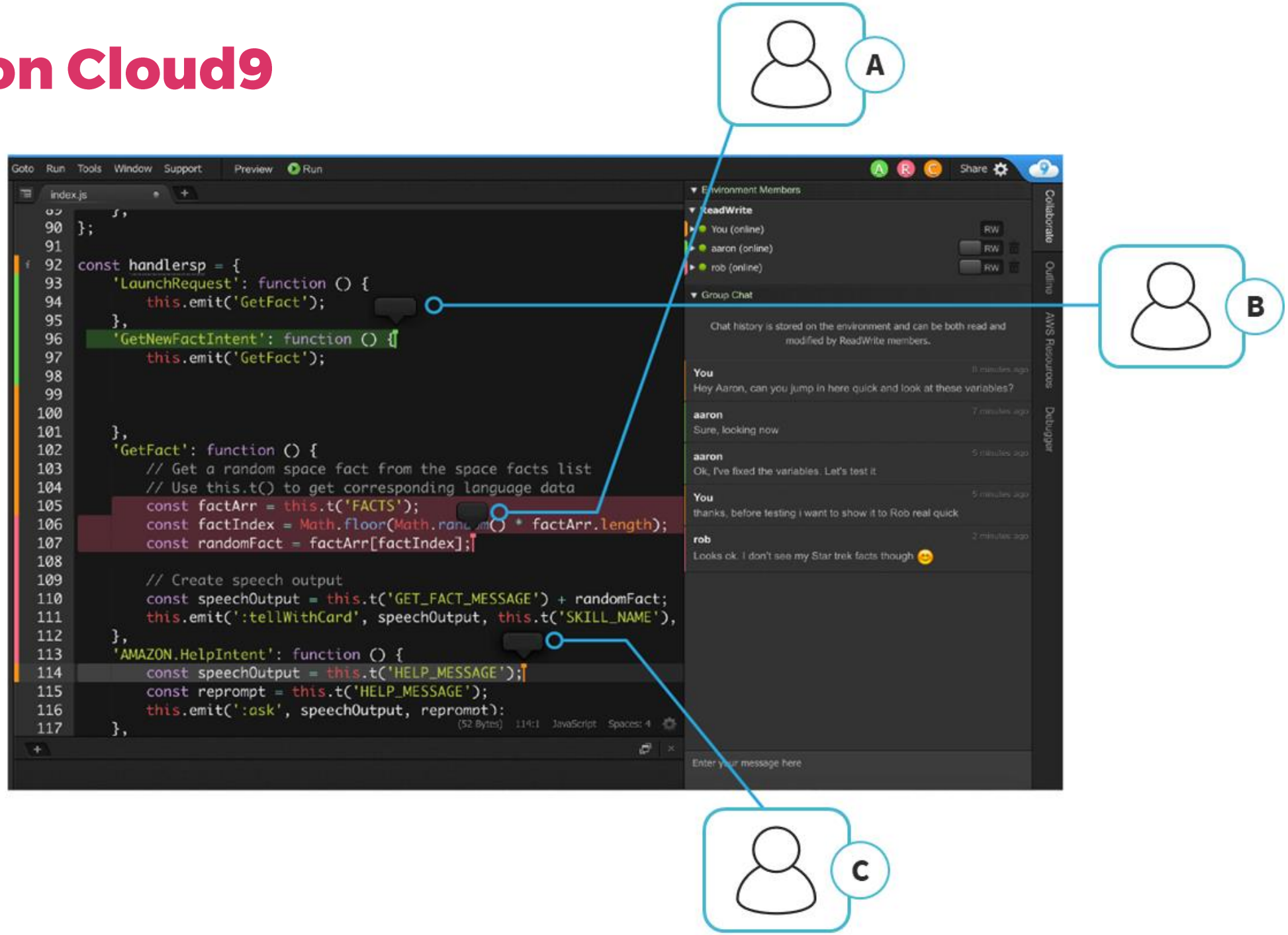


Khwaja, Amir A., Muniba Murtaza, and Hafiz F. Ahmed. "A Security Feature Framework for Programming Languages to Minimize Application Layer Vulnerabilities." Wiley Online Library. John Wiley & Sons, Ltd, November 7, 2019. https://onlinelibrary.wiley.com/doi/full/10.1002/spy2.95.

INTEGRATED DEVELOPMENT ENVIRONMENTS (IDEs

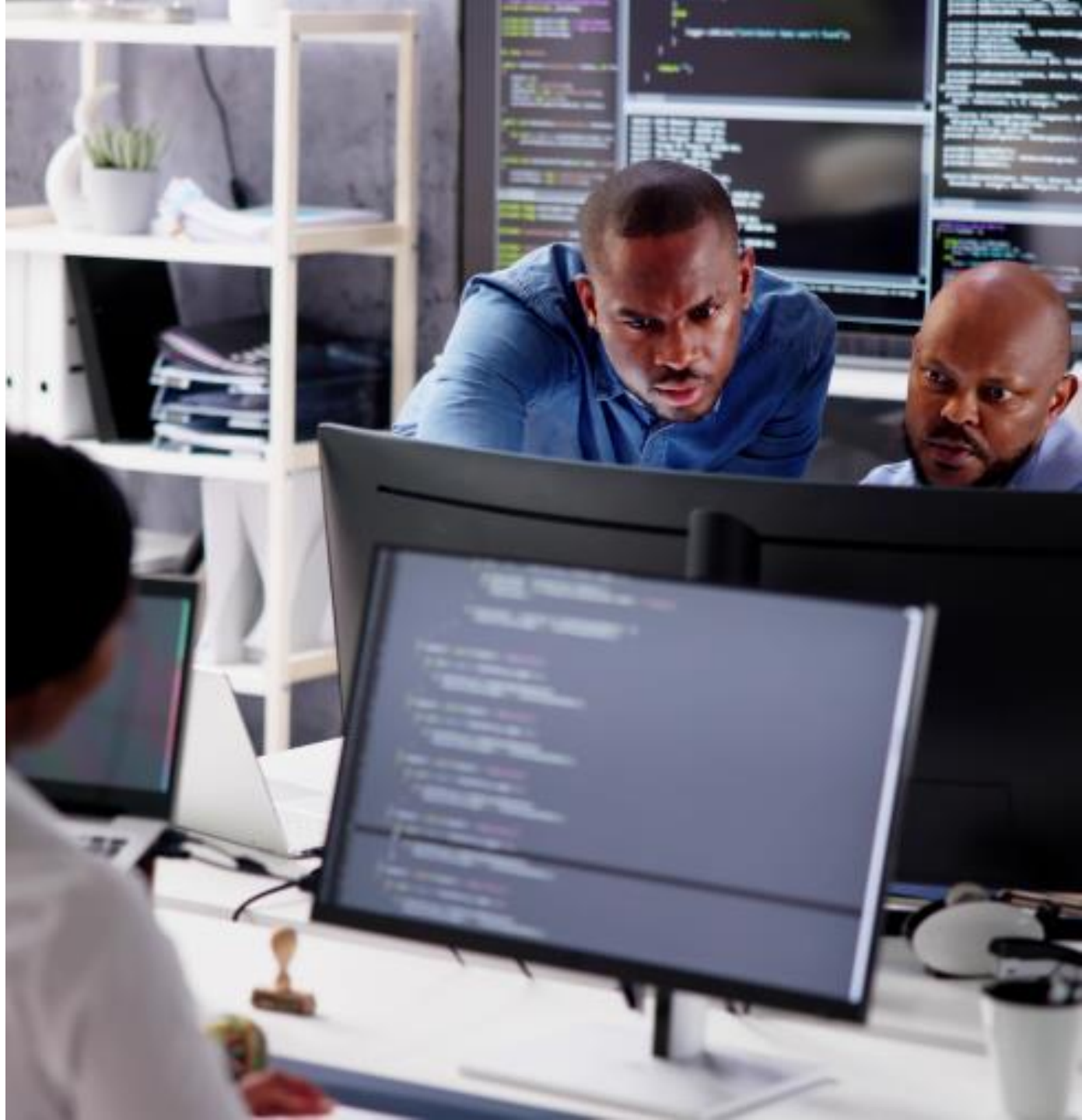
- **AWS Cloud9** is a cloud-based integrated development environment (IDE) that enables DevOps professionals to construct, run, and debug the code with only a browser
- It includes a code editor, debugger, and terminal
- Cloud9 comes prepackaged with vital tools for popular programming languages, including JavaScript, Python, and PHP
- There is no need to install files or configure the development machine for new projects

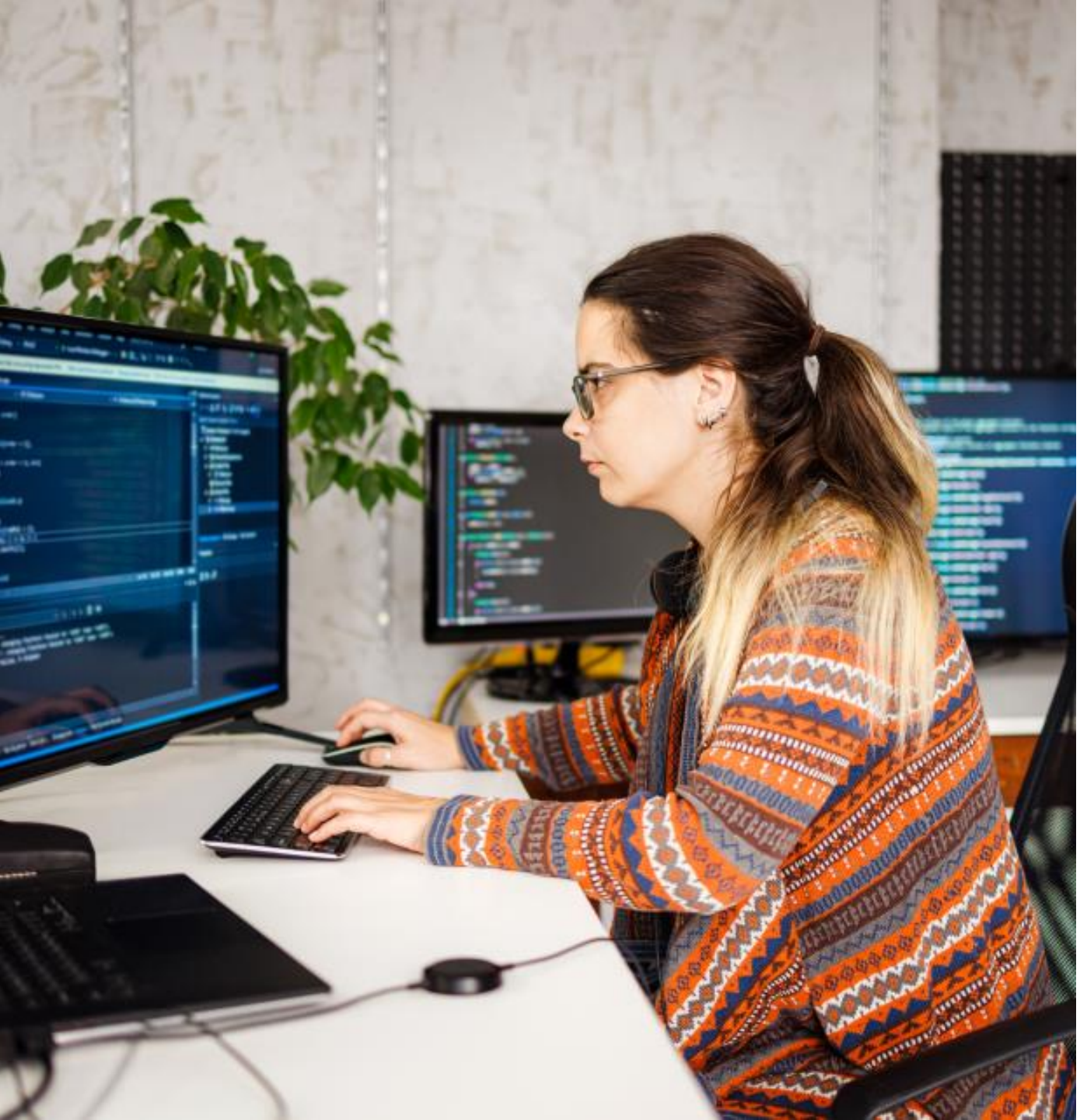
Amazon Cloud9



CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY (CI/CD)

- Continuous integration (CI) is a development technique that forces developers to integrate code into a shared repository several times a day
- Each check-in is then verified by an automated build, allowing teams to detect problems early
- The goal is to detect and locate bugs and security flaws quickly

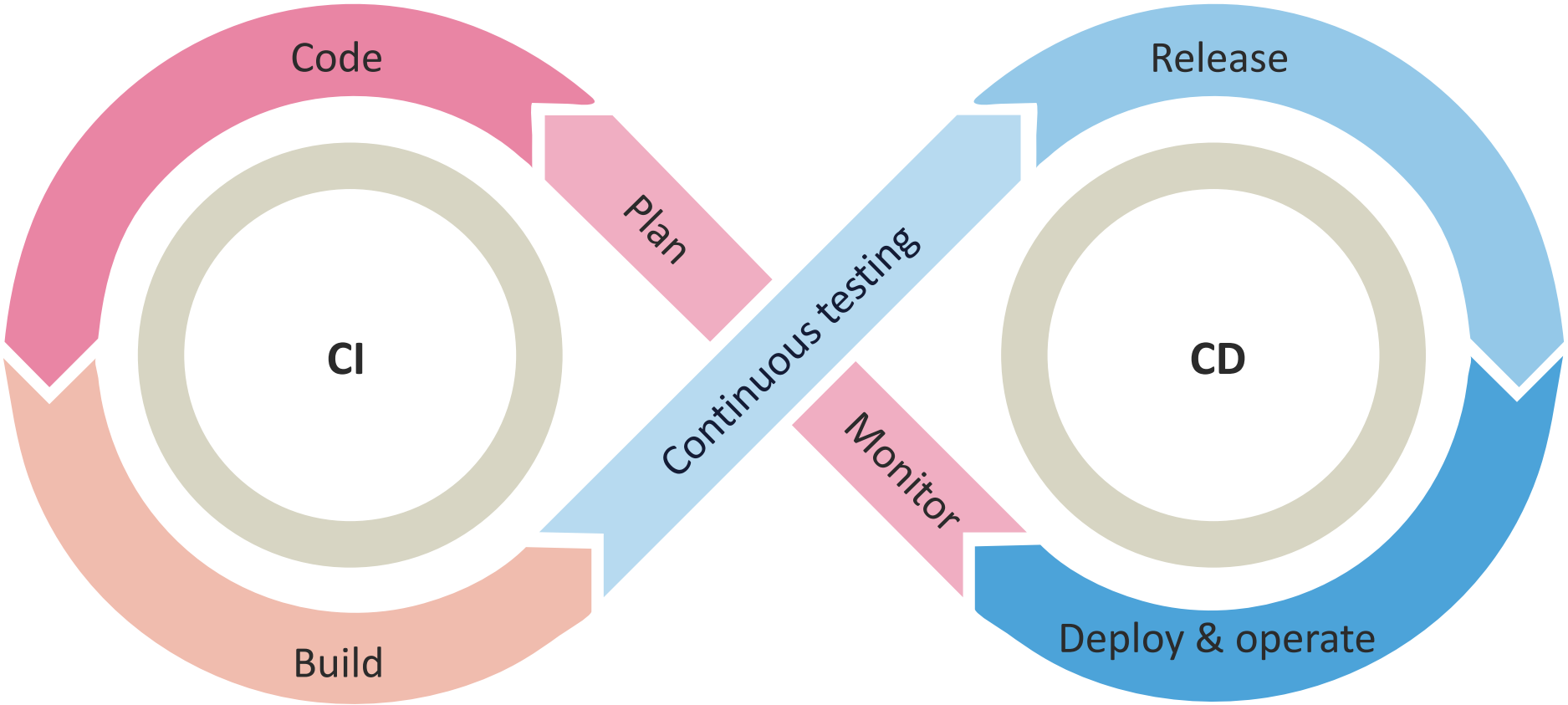




CI/CD

- Many development teams find that the CI/CD methodology significantly lessens integration challenges
- It enables the DevOps team to cultivate a cohesive software strategy more quickly
- Developers on AWS commonly use CI/CD

CI/CD

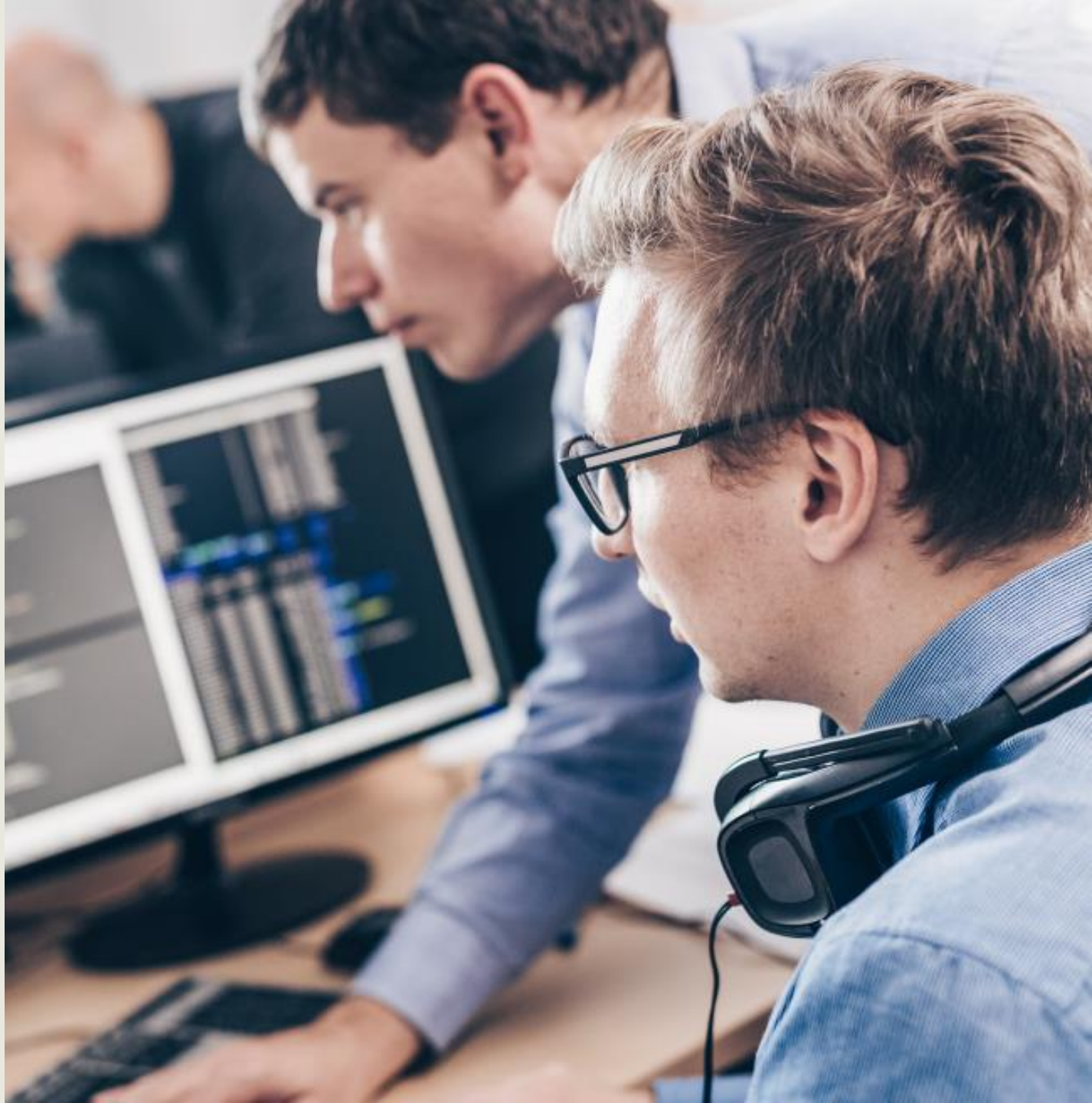

Code
Plan
Continuous testing
Release
CI
CD
Monitor
Build
Deploy & operate

SOFTWARE CONFIGURATION MANAGEMENT (SCM)

- Software configuration management (SCM) is a software engineering process to systematically manage, organize, and control the changes in the documents, codes, and other artifacts during the SDLC
- SCM is part of the cross-disciplinary field of configuration management (integrated product teams) and can correctly determine the revision history

REASONS TO USE SCM

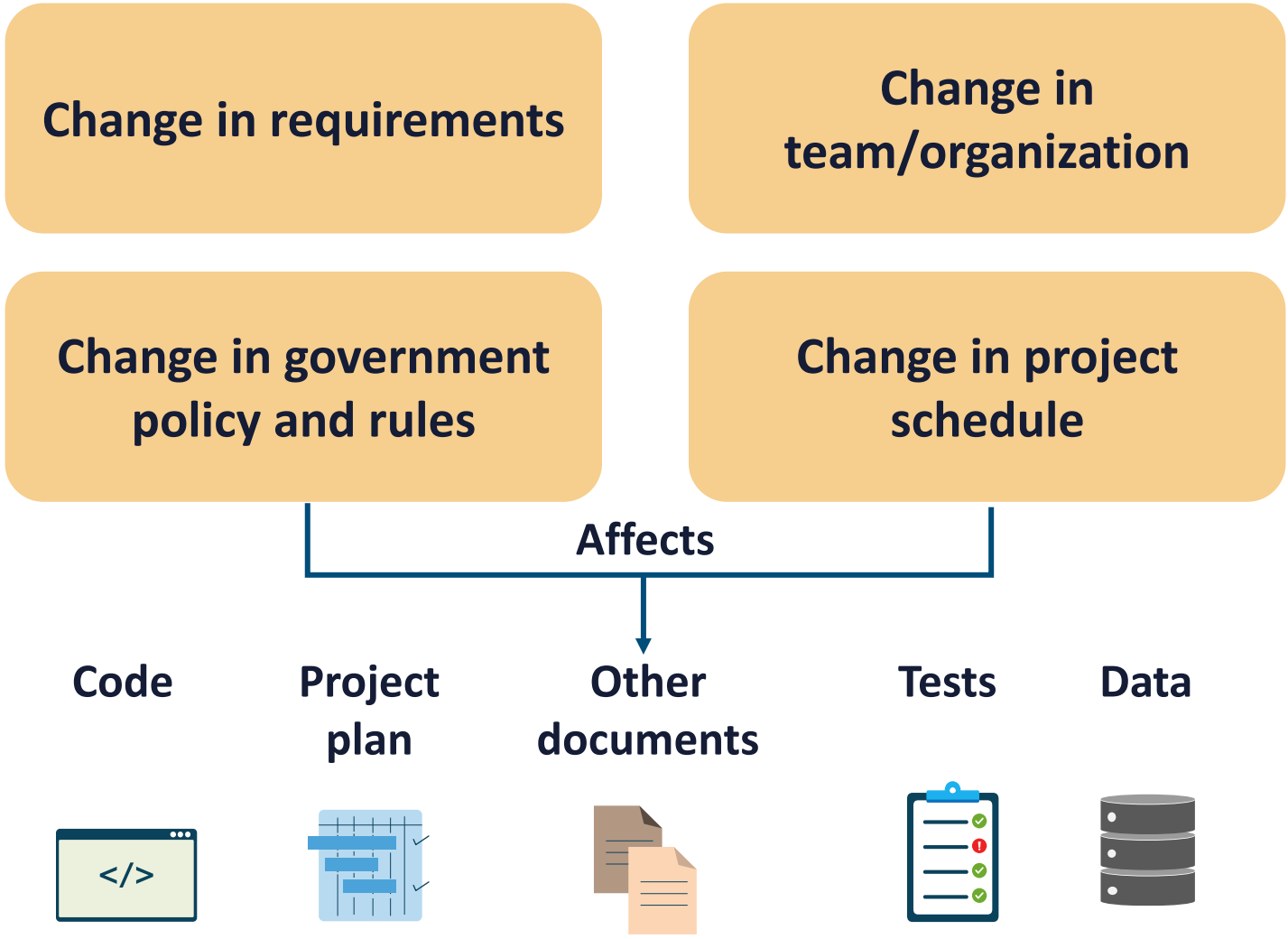
- To enhance productivity and minimize errors
- When there are several people working on applications that are continually updating (CI/CD or spiral development)
- If there are multiple versions, branches, microservices, and programmers involved in a software project, and the team is geographically dispersed yet is working concurrently



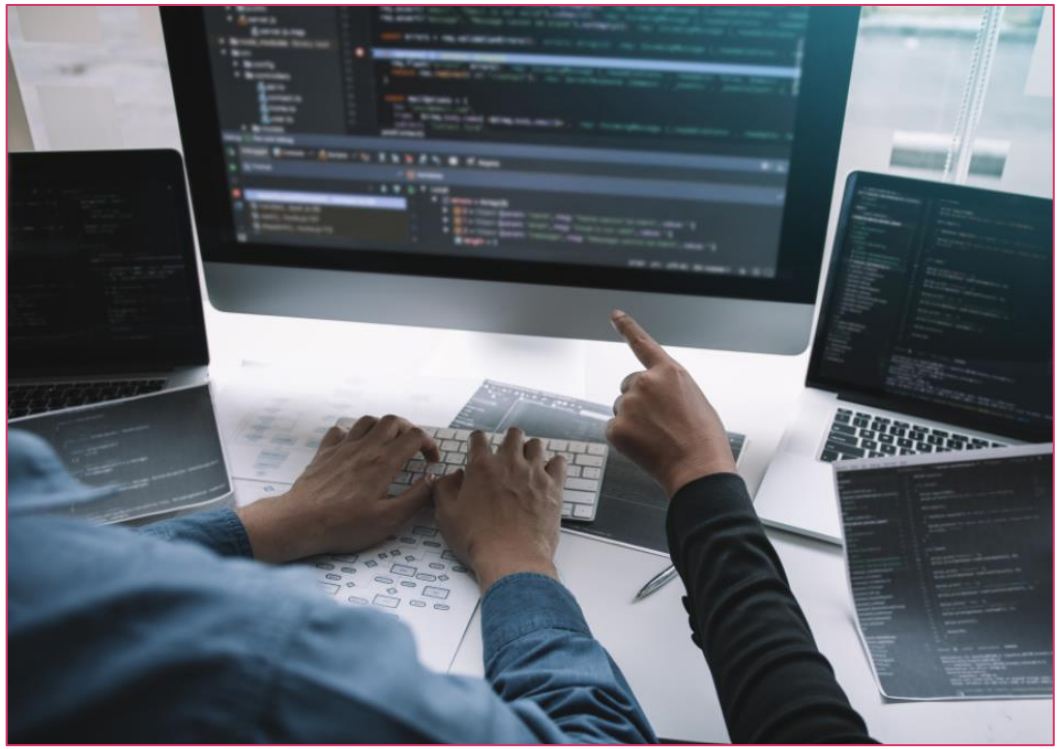
REASONS TO USE SCM

- Changes in customer requirements, policy, budget, and schedules need to be accommodated
- Software must be able to run on different platforms and operating systems
- There is a critical need to develop coordination among cross-functional stakeholders
- The costs involved in making changes to an app need to be controlled

SOFTWARE CONFIGURATION MANAGEMENT (SCM)



CODE REPOSITORY SECURITY

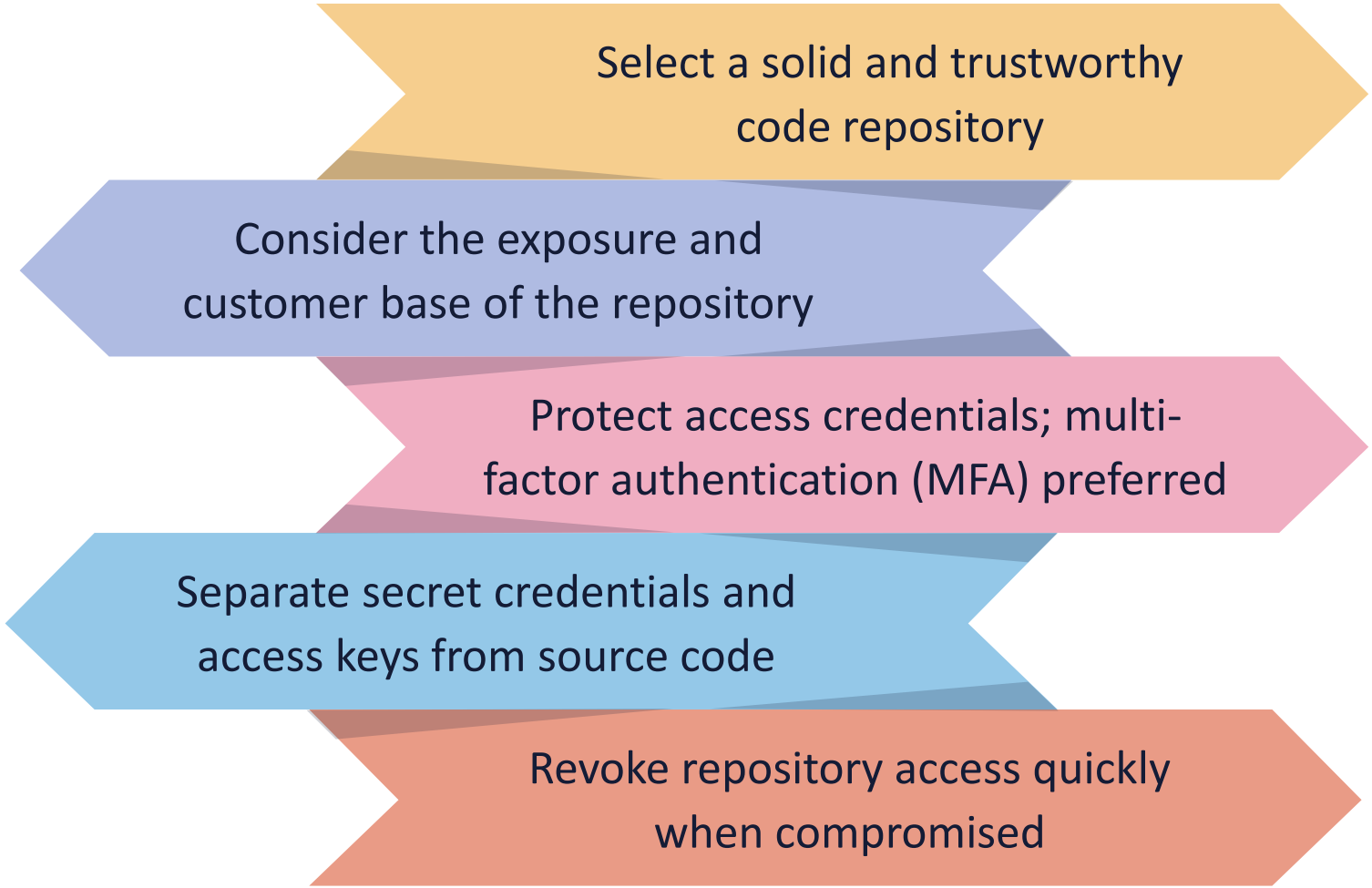


- Code is only as secure as the methods and systems used to generate it
- **Some of the advantages that come with using a secure code repository are version control, peer review, and built-in auditing**
- It is critical that the repository (such as GitHub or Azure) is an adequately secure central point of code storage and management

CODE REPOSITORY SECURITY

- Attackers can change a code base without your knowledge or permission due to loss/compromise of access credentials or breach of the core service
- If appropriate due diligence is applied to security measures, the benefits of using a code repository far outweigh the risks

CODE REPOSITORY SECURITY



STATIC APPLICATION SECURITY TESTING (SAST)

- SAST tools are also known as code analyzers that conduct a direct white-box analysis of the application source code
- The analysis runs on a static view of code, in that the code is not running at the time of the assessment

SAST

- SAST security tools are mainstream and are widely adopted throughout the software industry
- They have broad programming language support and use concepts that are relatively easy to comprehend
- SAST code analyzers have no visibility of the execution flow, can be slow, inaccurate, and outdated, and often need additional customization and/or tuning

DYNAMIC APPLICATION SECURITY TESTING (DAST)

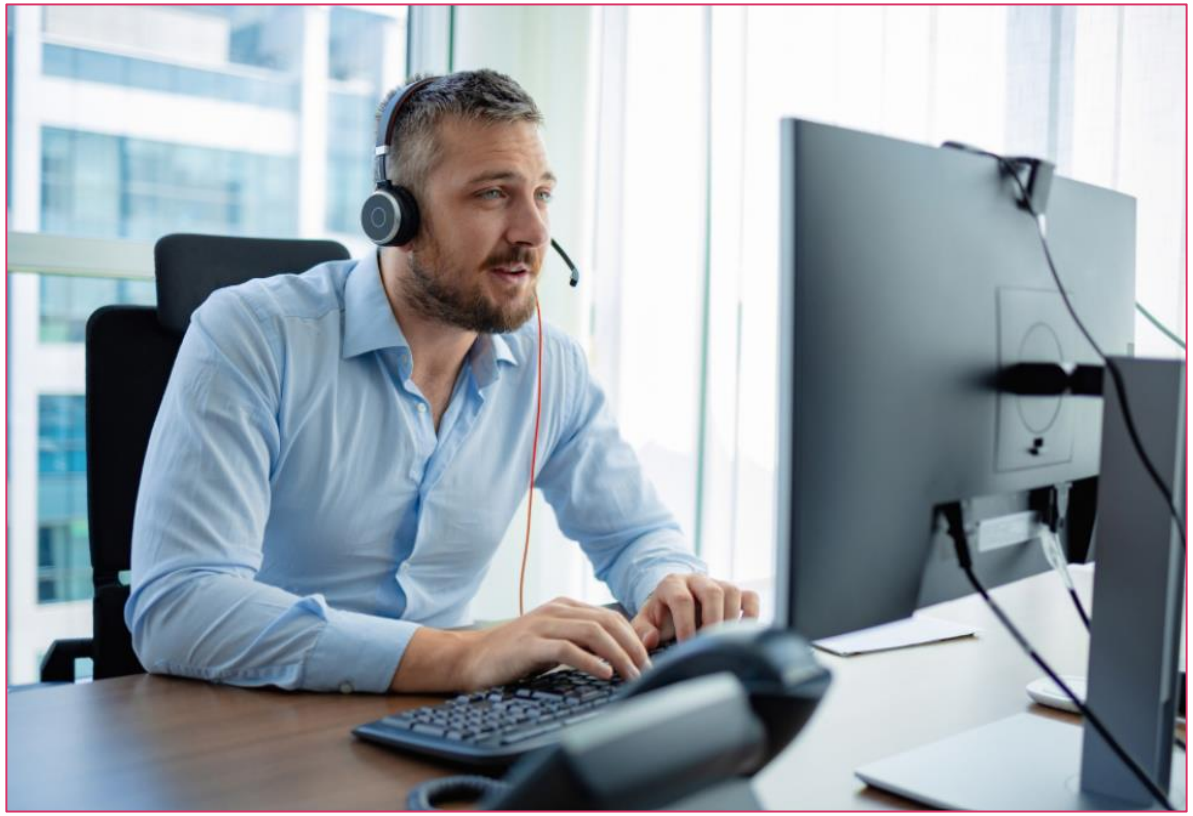
- DAST tools are most often web scanners like OWASP ZAP and Burp Suite (vulnerability scanners)
- When compared to SAST, they perform black-box analysis in that they do not have access to the code or the implementation specifics
- DAST only inspects the system's responses to a series of tests designed to highlight vulnerabilities

DAST

- DAST tools function independently of the underlying application platform and offer solid support for manual penetration testing
- A top-level DAST detects only ~20% of issues with no information provided on the location of the issue in the code base
- An experienced security background is necessary to interpret the results

INTERACTIVE APPLICATION SECURITY TESTING (IAST)

- IAST combines the advantages of SAST and DAST solutions
- It offers the benefits of a static view, because it can see the source code
- It also delivers the aids of a web scanner approach, since it sees the execution flow of the application during runtime
- It reportedly can detect ~100% of OWASP benchmarks in real-time with no false positives

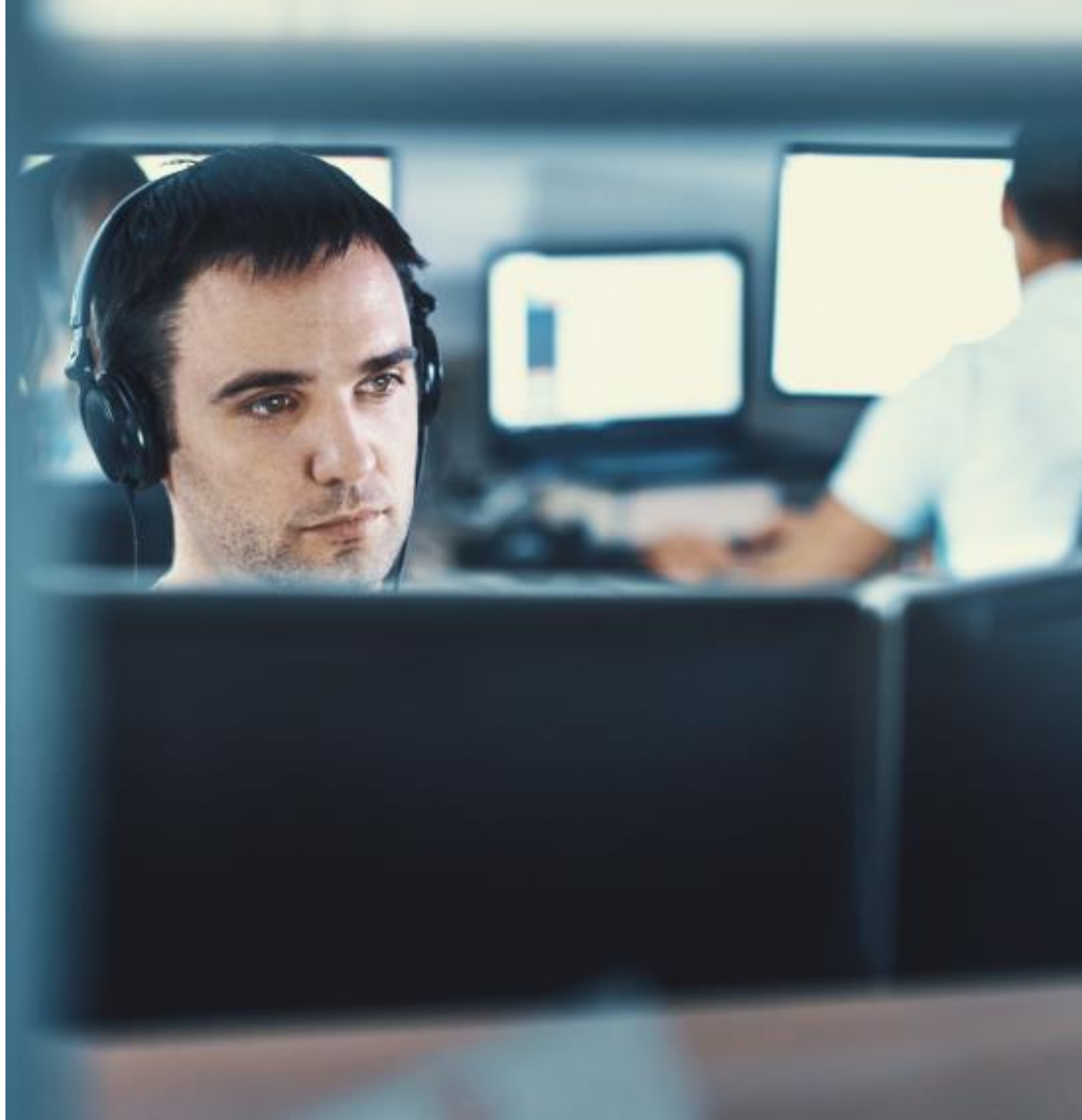


IAST

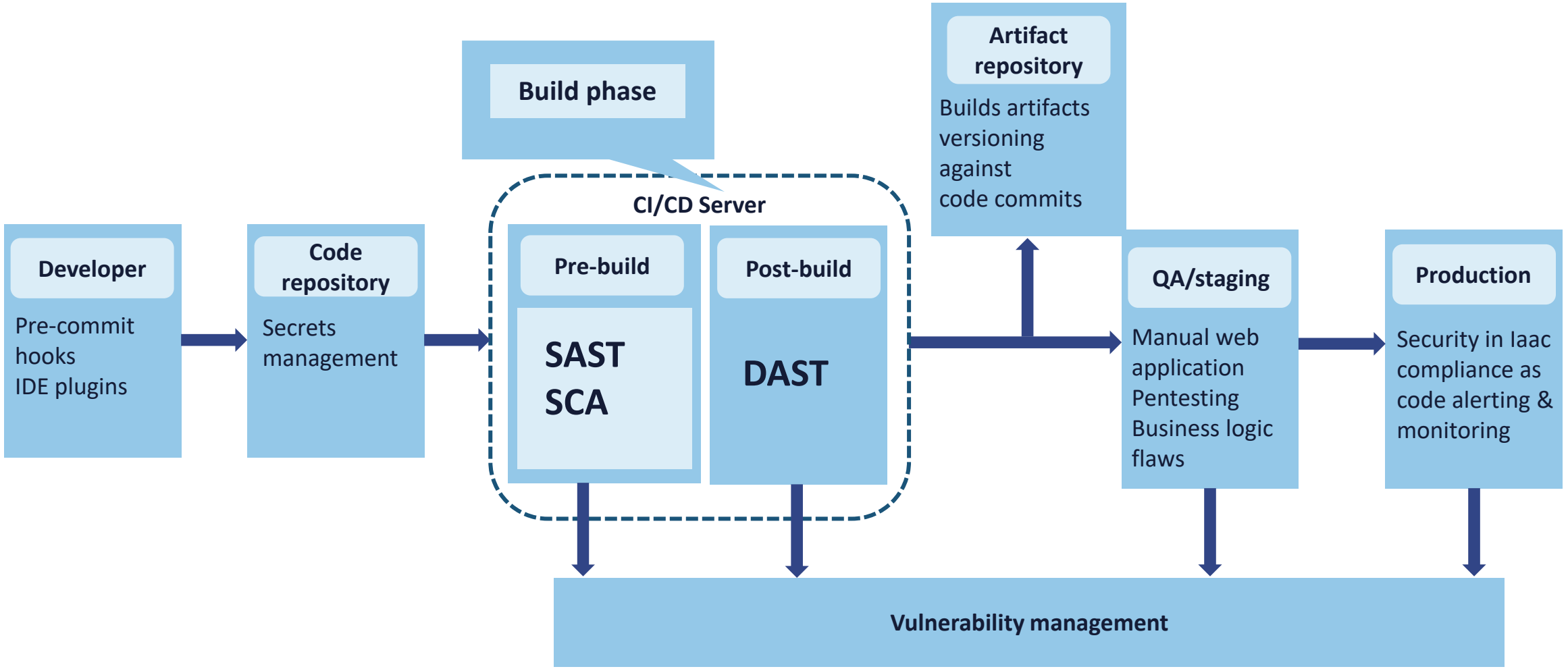
- IAST can flexibly be used in quality assurance (QA) and production environments, analyzing dependencies as well as legacy components
- There is no need to scan or attack the application
- IAST does continuous detection
- It is DevSecOps-friendly
- IAST integrates and communicates with task management systems to create unified workflows

SOFTWARE COMPOSITION ANALYSIS (SCA)

- SCA is a development technique for recognizing and controlling the open-source and third-party aspects of enterprise applications
- It assists teams in detecting app security vulnerabilities, licensing problems, or obsolete software components
- SCA also offers a software bill of materials (SBOM) to generate a comprehensive inventory of the open-source software packages and their roots



SOFTWARE COMPOSITION ANALYSIS



ASSESSING SOFTWARE SECURITY AND CODING GUIDELINES

Objectives

- Assess auditing, change logs, and risk analysis and mitigation
- Assess acquired software, managed services, and cloud services
- Describe source code vulnerabilities and API weaknesses in greater detail
- Outline secure coding practices
- Provide an overview of software-defined security (SDS)

ASSESSING AUDITING & LOGGING OF CHANGES

- Although the process of auditing produces a wide variety of results and supports accreditation and certification, the internal and external auditing tasks themselves must periodically be assessed
- A practical way to assess security auditors is to rely on continual third-party evaluations and certifications such as the Cybersecurity and Infrastructure Security Agency (CISA), Certified Internet Audit Professional (CIAP), ISACA Certified Information Security Manager (CISM), and Cloud Security Alliance (CSA) Certificate of Cloud Auditing Knowledge (CCAK)

ASSESSING AUDITING & LOGGING OF CHANGES

- When security information and event management (SIEM) and security orchestration, automation, and response (SOAR) systems are being deployed for centralized logging and automation, they must be fine-tuned for false positives and false negatives on a regular basis
- Auditors must go through regular training programs, conferences, seminars, and more for continuing education and advancement
- Security auditing is an art and a science where the practitioner only gets better with experience and improvement with tools

ASSESSING RISK ANALYSIS AND MITIGATION

- Performing risk assessment and analysis will never be a one-time program or initiative
- Many organizations will rely on ISO, ISACA, OWASP, ITIL 4, MITRE, and other architectures and frameworks to continually assess the success of risk management decisions
- These assessments may be conducted by steering committees, boards, or external accreditors and certifying bodies
- When assessing risk analysis and mitigation, often the decision will be made to transition away from legacy qualitative methodologies and move towards quantitative models such as Whitman or Open FAIR

GAP ANALYSIS

- A gap is the difference between the implemented existing controls and the predetermined control objectives
- Gap analysis is the outcome of corporate security strategy and governance
- **It is foundational for assessing the success of risk analysis and security control implementation**
- It provides the fundamental information security initiative action plans and programs

GAP ANALYSIS

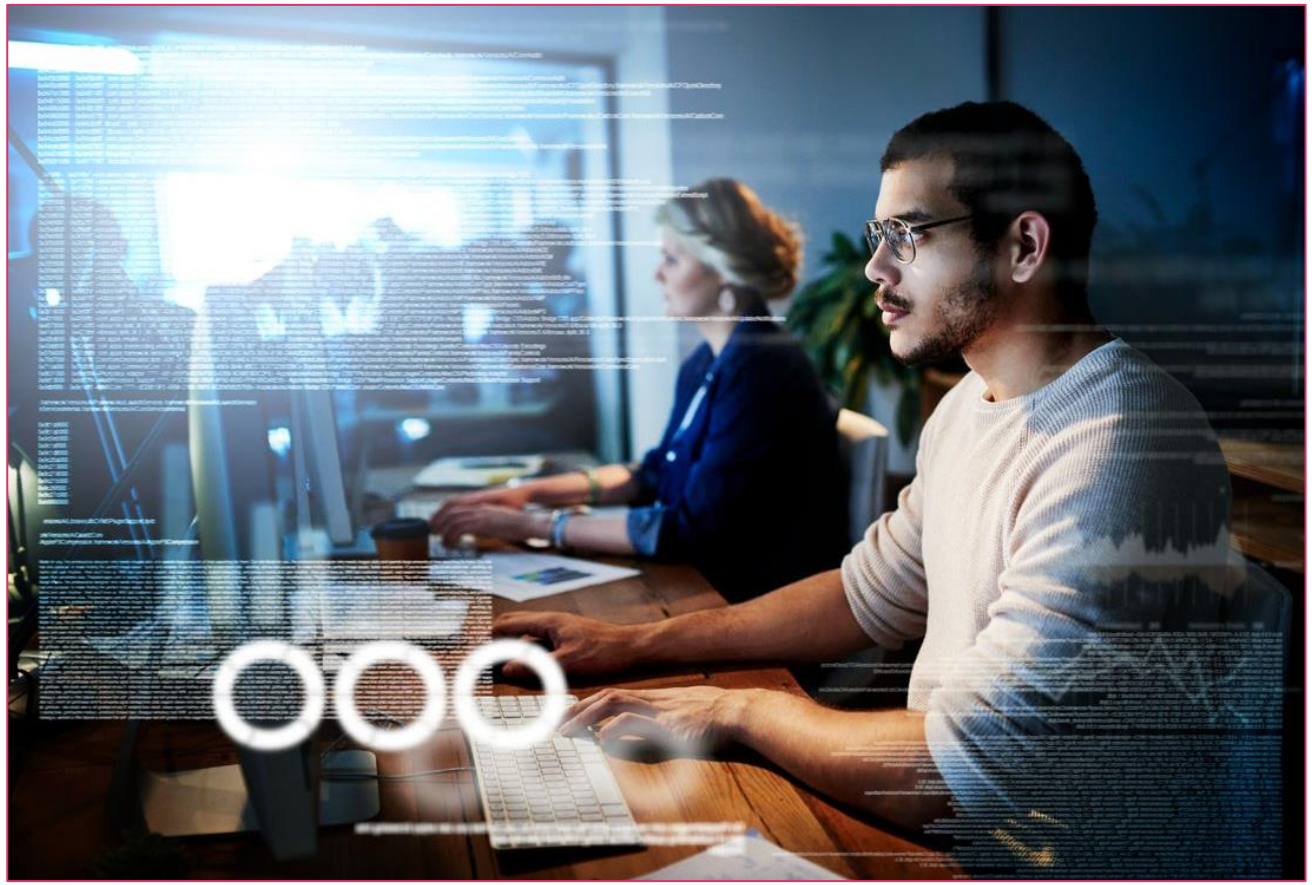
- Current countermeasures should be established according to the organization's risk appetite for each asset class
- Gap analysis should be performed in a reusable and repeatable manner to assess and report on the efficacy of executed controls
- Focus will be on established metrics such as key performance indicators (KPIs) and key goal indicators (KGIs)

ASSESSING ACQUIRED SOFTWARE: COTS

- Commercial-off-the-shelf (COTS) is a common pre-packaged (with varying licenses) and as-is solution
- COTS products are intended to be easily installed and to interoperate tightly with existing system components
- Almost all software bought by the public computer user fits into the COTS category (operating systems, office product suites, word processing, and email programs)

ASSESSING ACQUIRED SOFTWARE: MOTS

- MOTS stands for either modified or modifiable off-the-shelf software
- It is typically a product whose source code can be modified or customized by the purchaser, the vendor, or another party to meet the needs of the customer
- The evolution of Agile and CI/CD rapid development of containerized applications and microservices has led to more MOTS solutions
- **Modifications may be driven by the results of ongoing auditing and assessment of the acquired software solutions**



ASSESSING OPEN-SOURCE SOFTWARE SOLUTIONS

- Open-source software is a package whose code is accessible for public examination, alteration, and improvement
- **Many enterprises and products (90% by some estimates) use at least one open-source component, often without being aware of it**
- Normally, this software is built using public community collaboration and is preserved and updated on a voluntary basis
- Open-source software can be used according to a diversity of licenses, depending on what the developers have implemented

ASSESSING RISK ANALYSIS AND MITIGATION

- It is critical to evaluate and assess all open-source solutions because
  - Vulnerabilities are publicly known
  - There are no claims or obligations to be secure
  - Open-source software often includes or demands the use of vulnerable third-party libraries
  - There may be intellectual property challenges
  - There are over 200 types of licenses that can be used with open-source software
  - There is a general lack of warranty for its security, support, or content

ASSESSING MANAGED SERVICES

- Many enterprise solutions today are vendor-based or cloud-based Platform as a Service (PaaS) or Software as a Service (SaaS) solutions
- The key factor with these services is to perform proper due diligence and due care in formulating service-level agreements, master service agreements, and reciprocal agreements

ASSESSING MANAGED SERVICES

- There should be a regular review of security policies for remote access and single sign-on, federated access, and least privilege when accessing managed services with providers
- The CSA offers a Security Trust Assurance and Risk (STAR) registry with questionnaires filled out by many providers based on the Cloud Controls Matrix (CCM)

ASSESS CLOUD SERVICES WITH THE CCM

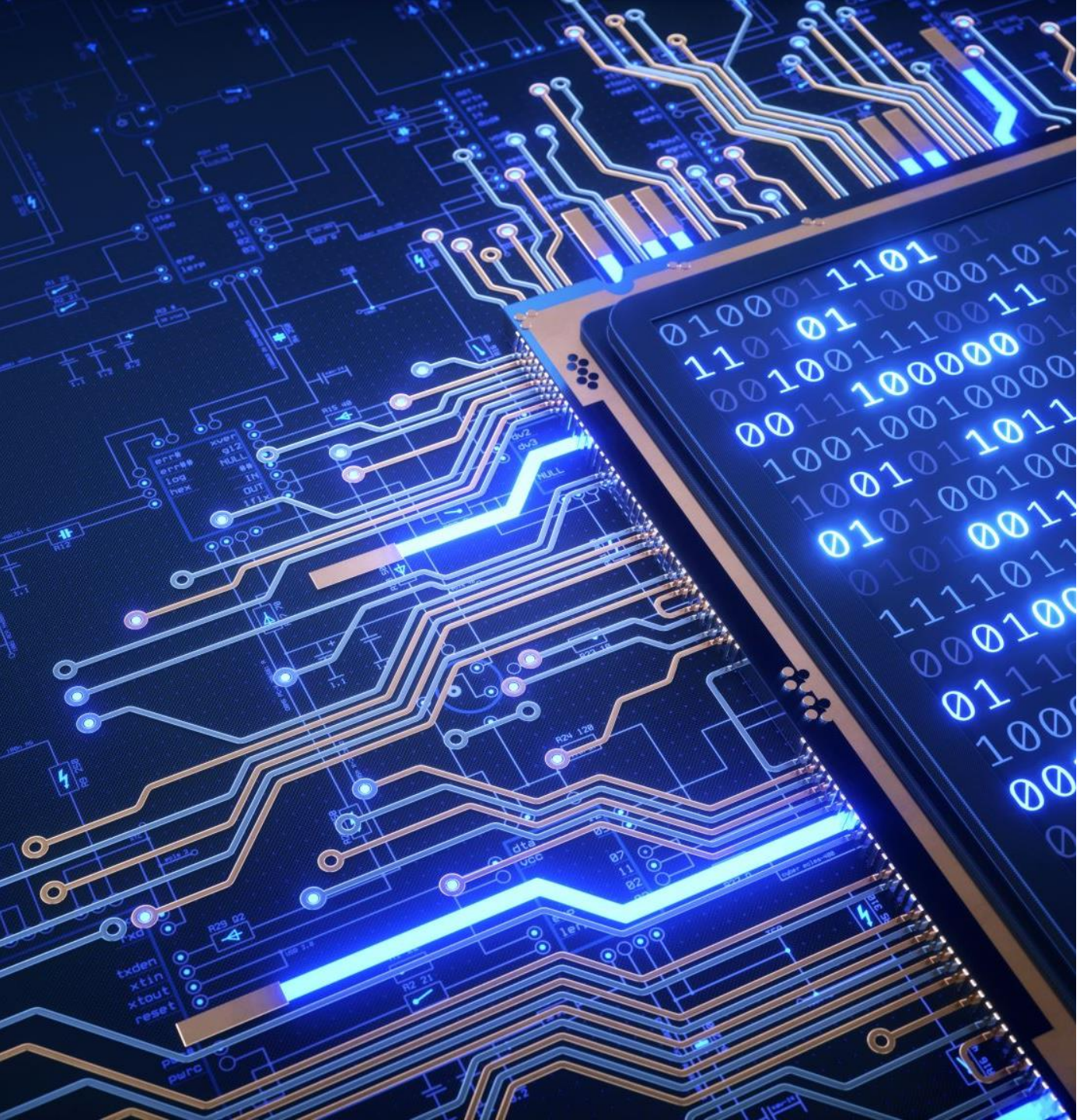
- The CCM is 197 control objectives structured in 17 domains covering all key aspects of cloud technology
- It is often used for the systematic assessment of a cloud implementation
- It offers guidance on which security controls should be implemented

ASSESS CLOUD SERVICES WITH THE CCM

- It can be used to audit and evaluate every participant and agent within the cloud supply chain
- It is considered the de facto standard for cloud security assurance and compliance
- The STAR Level 1: Security Questionnaire (CAIQ v4) offers an industry-accepted way to document what security controls exist in Infrastructure as a Service (IaaS), PaaS, and SaaS services

CCM V4 DOMAINS

- Application and Interface Security
- Audit and Assurance
- Business Continuity Mgmt & Op Resilience
- Change Control & Configuration Management
- Data Security & Privacy Lifecycle Management
- Datacenter Security



CCM V4 DOMAINS

- Cryptography, Encryption and Key Management
- Governance, Risk Management and Compliance
- Human Resources Security
- Identity & Access Management
- Security Infrastructure & Virtualization
- Interoperability & Portability

CCM V4 DOMAINS

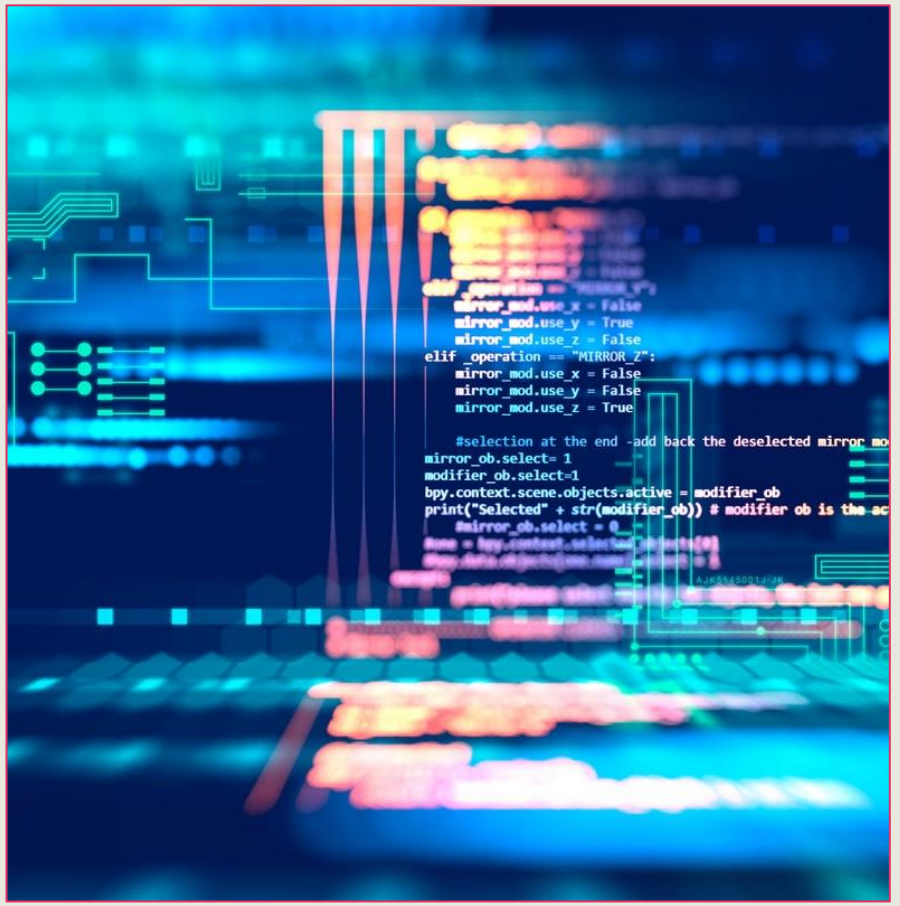
- Universal EndPoint Management
- Security Incident Management, E-Discovery & Cloud Forensics
- Supply Chain Management, Transparency & Accountability
- Threat & Vulnerability Management
- Logging and Monitoring

SOURCE CODE SECURITY WEAKNESSES AND VULNERABILITIES

- Code vulnerabilities are flaws in source code that, when exploited by attackers, will result in data leakage, code tampering, or possibly total data loss
- There are several well-established vulnerability databases and vendor support engines that keep programmers and development teams current on common weaknesses and attacks

EVAL INJECTION

- Eval injection involves referencing the improper neutralization of instructions in dynamically evaluated code
- This is one of the most critical code vulnerabilities that happens when a threat actor attempts to control some or all of an input string sent to an eval() function call
- The PHP eval() function is a fast way to execute string values as PHP code; however, when it is used with unknown inputs, it can render the code vulnerable to injection attacks
- Eval injection is an injection technique in which a malicious actor can inject a custom URL into the PHP eval() function

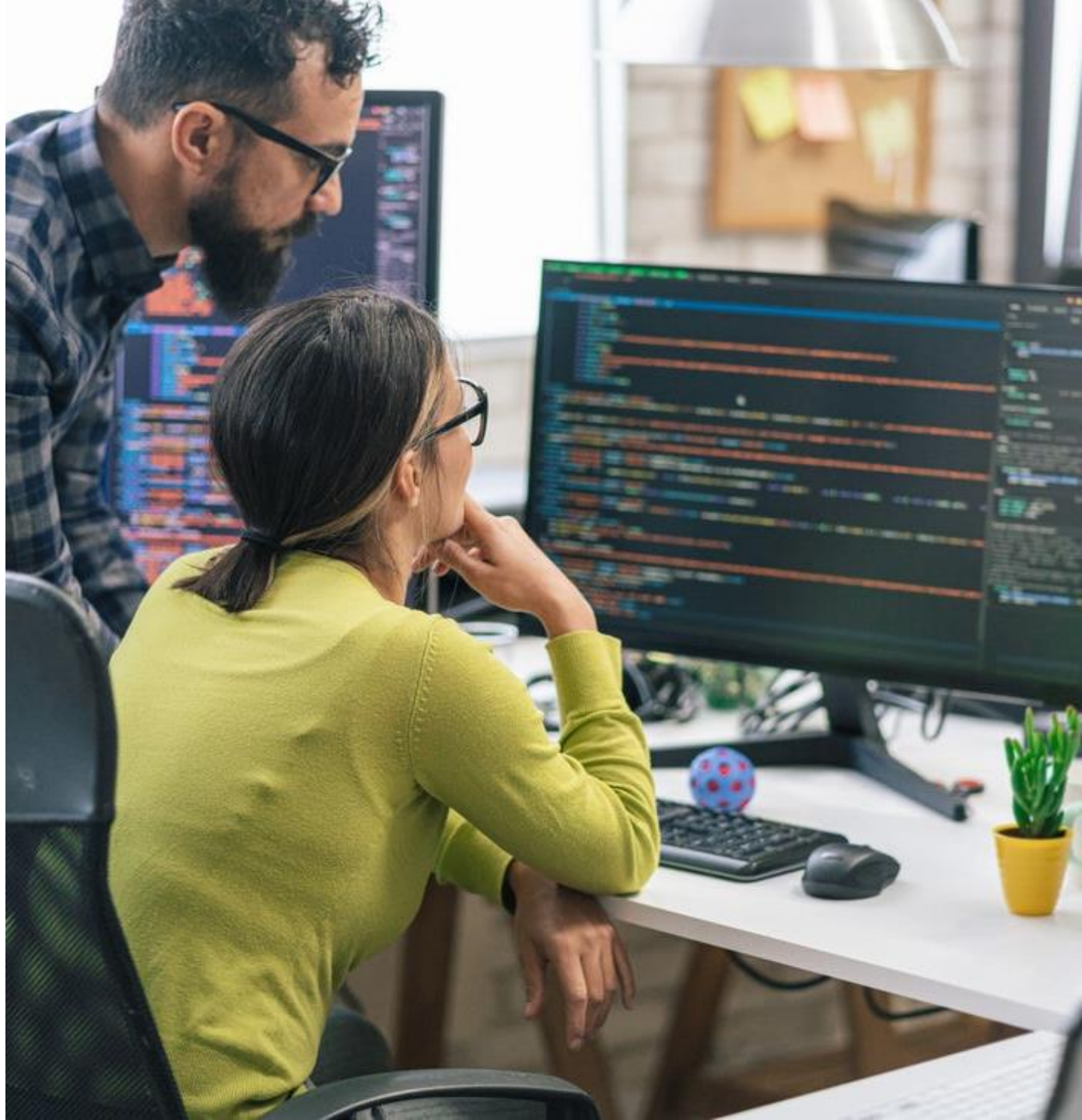


CROSS-SITE SCRIPTING (XSS)

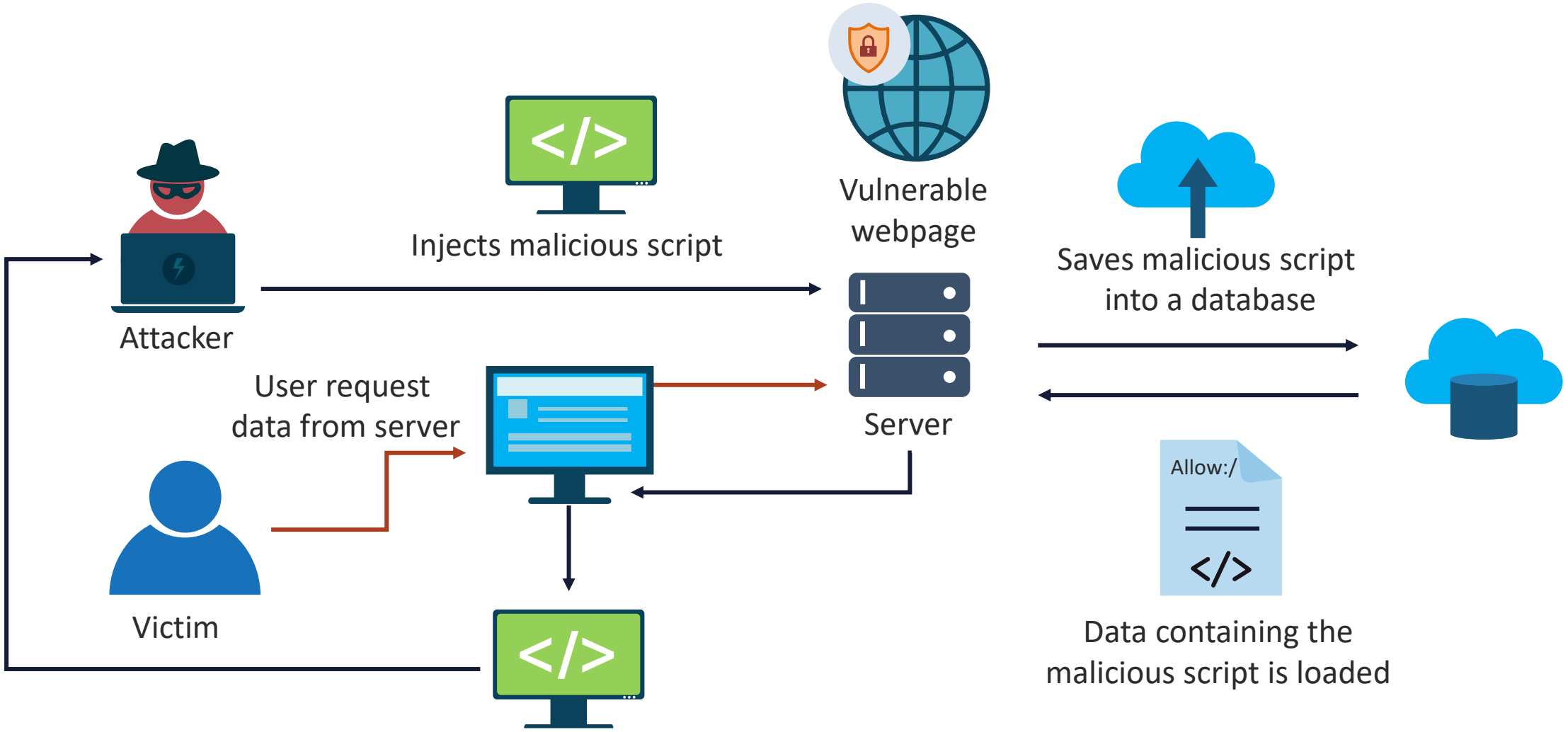
- XSS is considered the most serious code vulnerability
- It occurs when a malicious script, usually in HTML and JavaScript, is injected as data and attaches itself to rendered pages
- These scripts can be difficult to detect since they appear to come from a trusted source

XSS

- Often, this is performed in codes containing sensitive information such as credit card details or contact information
- With XSS, attackers can also insert special characters that cause the data to be interpreted as control information for the software
- This vulnerability enables certain application components to accept malicious commands and even conduct unauthorized actions



XSS ATTACKS


Vulnerable webpage
Injects malicious script
Attacker
Saves malicious script into a database
User request data from server
Server
Allow:/
Victim
Data containing the malicious script is loaded
Malicious script may get executed and call back to the attacker

SOURCE CODE SECURITY VULNERABILITIES

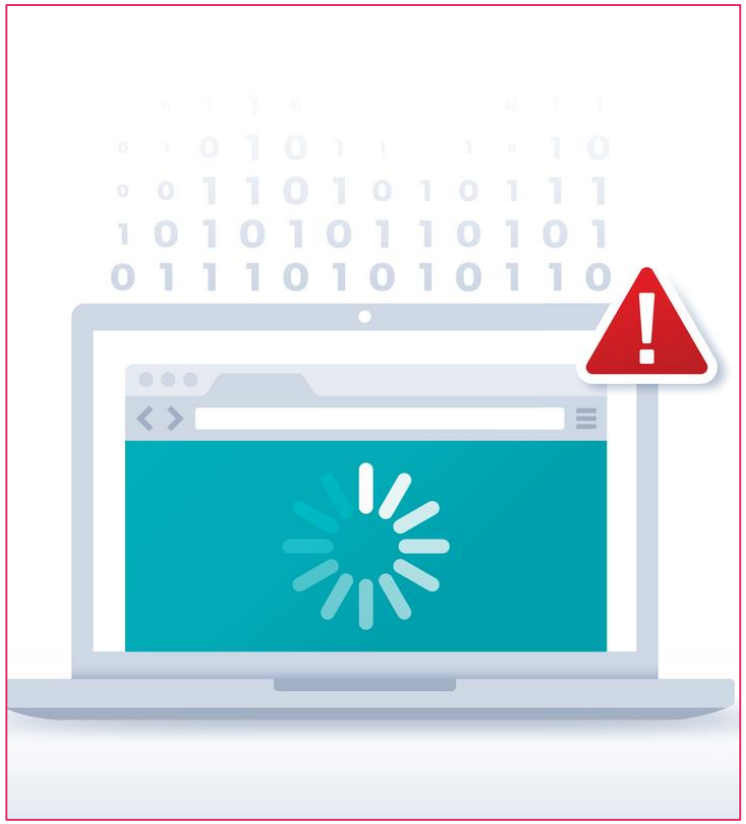
- Utilizing **hard-coded (embedded) credentials/keys** is considered an unsecured coding practice and can lead to critical issues
- For several reasons, coders tend to still use **weak encryption algorithms and cryptographic hashes** in their solutions, often for performance reasons
- Using standard **pseudo-random number generators** leaves one vulnerable to cryptographic attacks:
  - These leave software prone to insecure randomness errors, which happen when a function producing predictable values is used as a source of randomness

ASSESSING API SECURITY WITH OWASP TOP 10 2023

- **API1:2023 – Broken Object Level Authorization:**
  - APIs tend to expose endpoints that handle object identifiers so object-level authorization checks should be considered in every function that accesses a data source using an ID from the user
- **API2:2023 – Broken Authentication:**
  - Authentication methods are often deployed incorrectly, permitting attackers to compromise authentication tokens or exploit implementation flaws
- **API3:2023 – Broken Object Property Level Authorization:**
  - API3 combines the previous versions of API3:2019 – Excessive Data Exposure and API6:2019 – Mass Assignment
  - A lack of or improper authorization validation at the object property level can result in data exposure or manipulation by unauthorized parties

ASSESSING API SECURITY WITH OWASP TOP 10 2023

- **API4:2023 – Unrestricted Resource Consumption:**
  - Meeting API requests demands resources such as network bandwidth, CPU, memory, and storage
- **API5:2023 – Broken Function Level Authorization:**
  - Authorization flaws often occur with intricate access control policies that involve different hierarchies, groups, and roles
  - There can also be an unclear separation between administrative and normal functions
- **API6:2023 – Unrestricted Access to Sensitive Business Flows:**
  - APIs may expose data flows such as buying a ticket or posting a comment, without considering how the functionality might damage the business if done excessively in an automated way

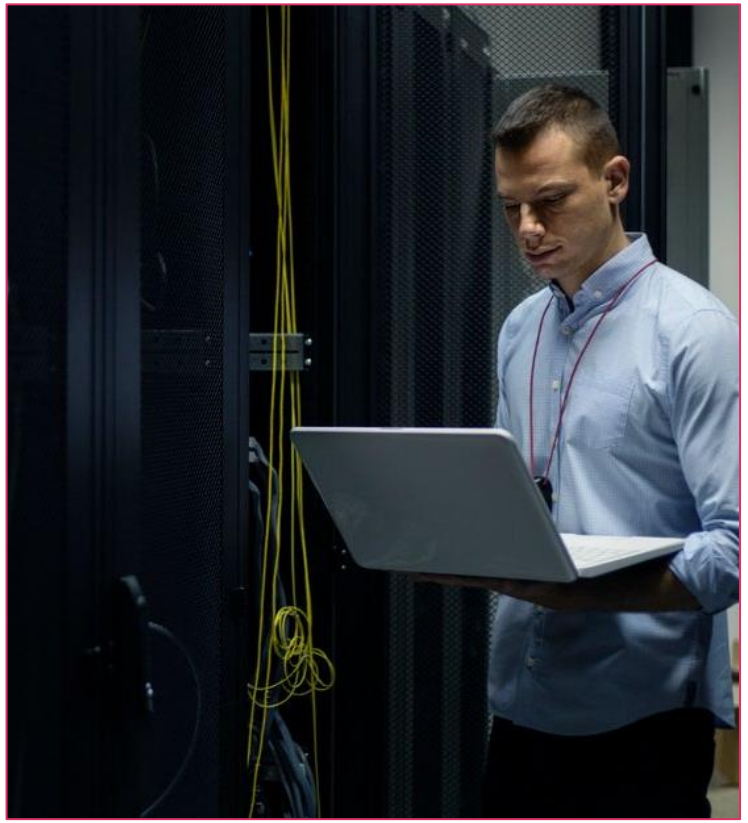


ASSESSING API SECURITY WITH OWASP TOP 10 2023

- **API7:2023 – Server-Side Request Forgery:**
  - A Server-Side Request Forgery (SSRF) can happen when an API is fetching a remote resource without validating the user-supplied URI
  - This lets an attacker trick the application into sending a fashioned request to an unexpected target, even through a firewall or a VPN
- **API8:2023 – Security Misconfiguration:**
  - APIs and the systems that support them usually have complicated configurations, intended to make the APIs more customizable

ASSESSING API SECURITY WITH OWASP TOP 10 2023

- **API9:2023 – Improper Inventory Management:**
  - APIs may expose more endpoints than legacy web applications
  - This fact makes proper and updated documentation extremely vital
  - The complete inventory of hosts and deployed API versions is important to alleviate issues like deprecated API versions and exposed debug endpoints
- **API10:2023 – Unsafe Consumption of APIs:**
  - Developers tend to trust data received from third-party APIs more than input from users
  - This can lead to using weaker security standards
  - To compromise APIs, attackers focus on integrated third-party services instead of directly attacking the target API

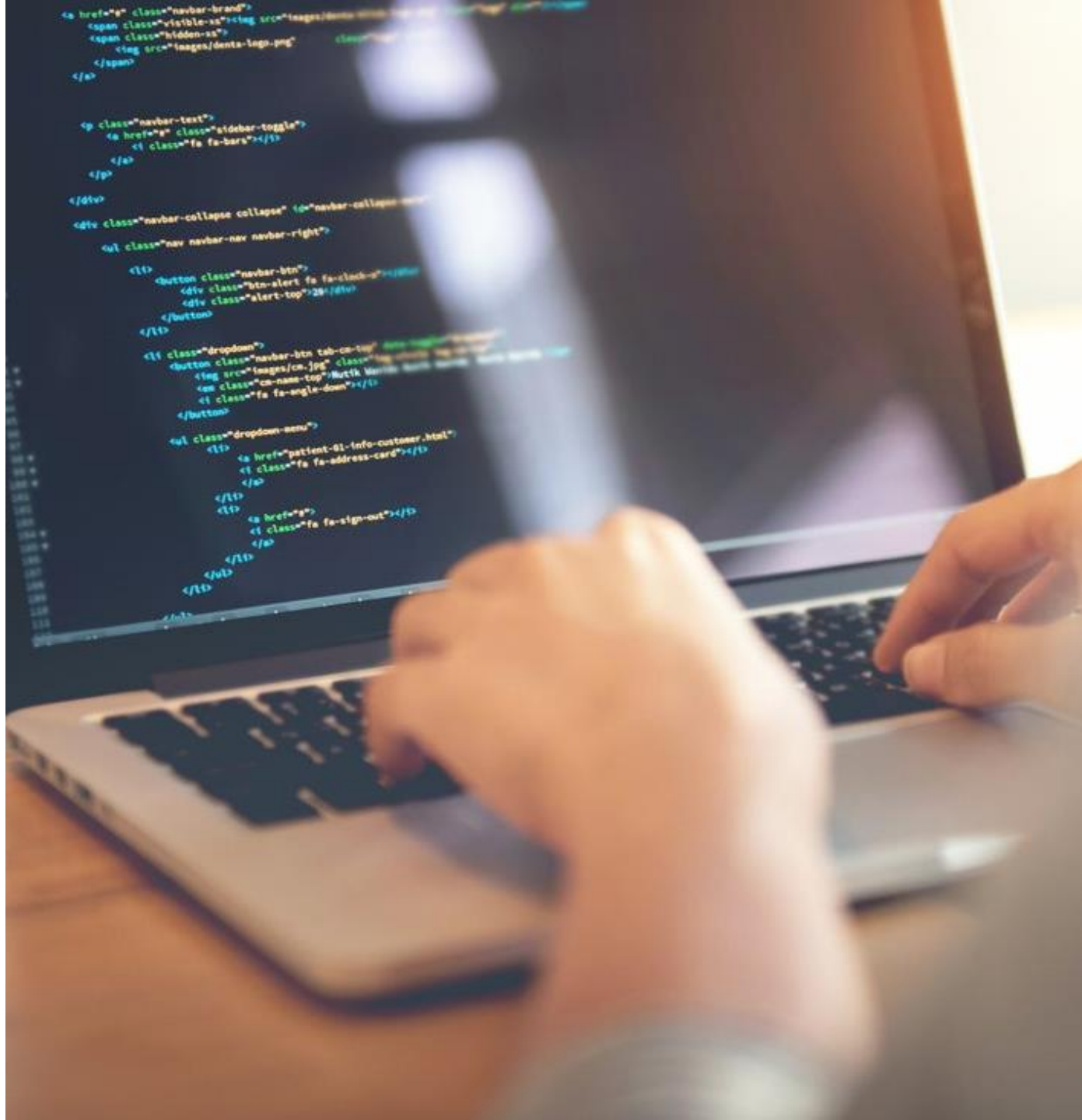


OWASP ENTERPRISE SECURITY API (ESAPI) PROJECT

- ESAPI is a free and open-source web application security control library that empowers programmers to construct lower-risk applications and interfaces
- The ESAPI libraries are intended to make it stress-free for developers to retrofit security into existing applications
- It offers a collection of security control interfaces that define various types of parameters passed to security controls

SECURE CODING PRACTICES

- Clearly define roles and responsibilities for programmers and other stakeholders in the coding processes
- Offer development team leads and members with sufficient software security training
- Implement a secure software development lifecycle (DevSecOps with Agile or CI/CD)
- Institute secure coding standards:
  - Input validation
  - Proper error and exception handling
  - Reusable container microservices that are secure by default

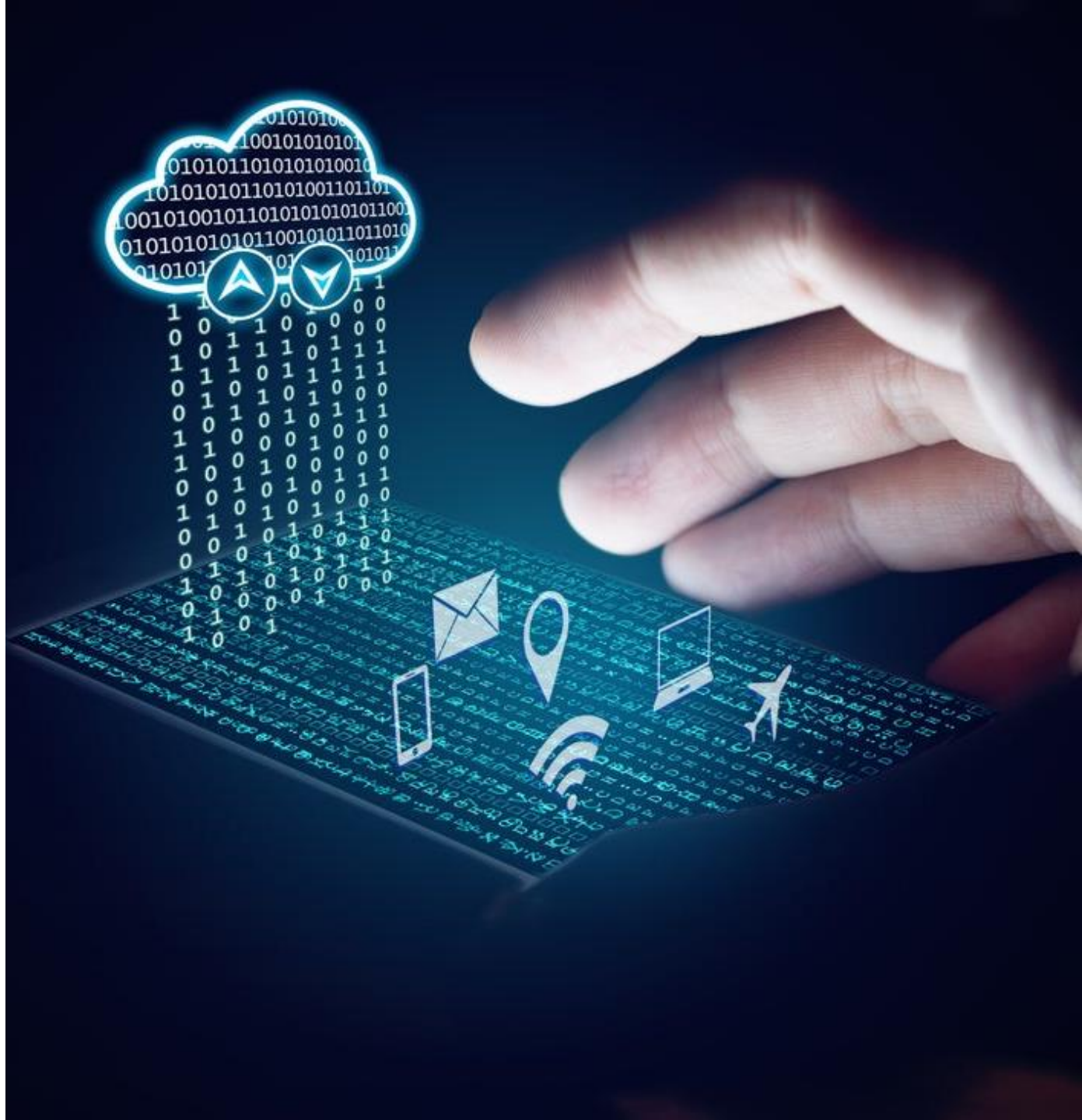


SECURE CODING PRACTICES

- Leverage existing guidance, such as the OWASP Developer Guide project
- Construct a reusable and secure object library
- Verify the effectiveness of security controls
- Institute secure outsourced programming habits, including defining security requirements and verification methodologies in both the request for proposal (RFP) and contract processes
- Consider using the OWASP Application Security Verification Standard (ASVS) project

OWASP APPLICATION SECURITY VERIFICATION STANDARD (ASVS)

- The ASVS project gives developers a list of requirements for secure development
- The standard offers a basis for testing application technical security controls, as well as any technical security controls in the environment, that are used to mitigate attacks, such as XSS and SQL injection
- The requirements were developed to be used:
  - As a meaningful metric
  - As secure development guidance
  - During the procurement process



SOFTWARE-DEFINED SECURITY (SDS)

- SDS is a model in which information security is highly controlled, often using virtualization
- The functionality of network security devices, such as next-gen firewalls, intrusion detection and prevention, identity and access controls, and network segmentation, are removed from hardware devices to a software layer
- SDS exploits the software-defined networking (SDN) initiative to enhance network security

SDS ADVANTAGES

- Offers resourceful and dynamic countermeasures to security attacks
- Separates security away from traditional hardware vulnerabilities
- Ability to dynamically configure existing network nodes allows for rapid attack mitigation from zero-day attacks
- Provides a synchronized view of logical security policies within the SDN controller model (not tied to any server or specialized security device)

SDS ADVANTAGES

- Provides visibility of information from one source
- Integrates with emerging technology to correlate events in a simpler way and respond more efficiently and intelligently to threats
- Enables centralized management of security, which is implemented, controlled, and managed by security software through the SDN controller
- Facilitates Internet of Things (IoT) and Bring Your Own Device (BYOD) connectivity and security

SDS FRAMEWORK