Overview

Level	Total Activities	Completed	Completion
1	25	11	44%
2	56	2	4%
3	62	0	0%
4	30	0	0%

'—' - No progress | '◖' - Partly Implemented | '✓' - Fully Implemented

Build and Deployment

Build

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
1	Defined build process	A *build process*include more than just compiling your source code. It also includes steps such as managing (third party) dependencies, environment configuration, running the...	—	◖	◖	✓
2	Building and testing of artifacts in virtual environments	While building and testing artifacts, third party systems, application frameworks and 3rd party libraries are used. These might be malicious as a result of vulnerable...	—	—	—	—
2	Pinning of artifacts		◖	◖	◖	—
2	SBOM of components	SBOM (Software Bill of Materials) is a document that lists all components, libraries, and dependencies used in a software application or container image. Creating an...	—	—	—	—
3	Signing of code		✓	—	—	◖

Deployment

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
1	Defined deployment process		✓	✓	✓	✓
1	Inventory of production components		—	—	—	—
2	Defined decommissioning process	The decommissioning process in the context of Docker and Kubernetes involves retiring Docker containers, images, and Kubernetes resources that are no longer needed or have...	✓	✓	✓	✓
2	Environment depending configuration parameters (secrets)		—	—	—	—
2	Evaluation of the trust of used components		—	—	—	—
2	Inventory of production artifacts		—	—	—	—
3	Handover of confidential parameters		—	—	—	—
3	Inventory of production dependencies		—	—	—	—
3	Rolling update on deployment		—	—	—	—
4	Same artifact for environments		—	—	—	—
4	Usage of feature toggles		—	—	—	—

Patch Management

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
1	A patch policy is defined		✓	✓	✓	✓
1	Automated PRs for patches		✓	✓	✓	✓
2	Automated merge of automated PRs	Automated merges of automated created PRs for outdated dependencies.	✓	✓	✓	✓
2	Nightly build of images (base images)	A base image is a pre-built image that serves as a starting point for building new images or containers. These base images usually include an…	—	—	—	—
2	Reduction of the attack surface	Distroless images are minimal, stripped-down base images that contain only the essential components required to run your application. They do not include package managers, shells,…	—	—	—	—
2	Usage of a maximum lifetime for images	The maximum lifetime for a Docker container refers to the duration a container should be allowed to run before it is considered outdated, stale, or…	—	—	—	—
3	Automated deployment of automated PRs	Automated merges of automated created PRs for outdated dependencies.	✓	◖	—	—
4	Usage of a short maximum lifetime for images	The maximum lifetime for a Docker container refers to the duration a container should be allowed to run before it is considered outdated, stale, or…	—	—	—	—

Culture and Organization

Design

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
1	Conduction of simple threat modeling on technical level	OWASP SAMM Description Threat modeling is a structured activity for identifying, evaluating, and managing system threats, architectural design flaws, and recommended security mitigations. It is…	✓	✓	✓	✓
2	Information security targets are communicated		—	—	—	—
3	Conduction of simple threat modeling on business level		—	—	—	—
3	Creation of simple abuse stories		—	—	—	—
3	Creation of threat modeling processes and standards		—	—	—	—
4	Conduction of advanced threat modeling	**Example High Maturity Scenario:** Based on a detailed threat model defined and updated through code, the team decides the following: • Local encrypted caches need to…	—	—	—	—

Education and Guidance

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
1	Ad-Hoc Security trainings for software developers		✓	✓	✓	✓
1	Security consulting on request		—	—	—	—
2	Each team has a security champion	Implement a program where each software development team has a member considered a "Security Champion" who is the liaison between Information Security and developers. Depending…	—	—	—	—
2	Regular security training for all	Conduct security awareness training for all roles currently involved in the management, development, testing, or auditing of the software. The goal is to increase the…	—	—	—	—
2	Regular security training of security champions		—	—	—	—
2	Reward of good communication		—	—	—	—
2	Security code review	Benefits • New vulnerabilities may be found before reaching production. • Old vulnerabilities are found and fixed.	—	—	—	—
3	Conduction of build-it, break-it, fix-it contests		◖	—	—	—
3	Office Hours		—	—	—	—
3	Security Coaching		—	—	—	—
3	Security-Lessoned-Learned		—	—	—	—
3	Simple mob hacking	Guidelines for your simple mob hacking session • All exploits happen via the user interface. • No need for security/hacking tools. • No need for deep technical or…	—	—	—	—
4	Aligning security in teams		—	—	—	—
4	Conduction of collaborative team security checks		—	—	—	—
4	Conduction of war games		—	—	—	—
4	Regular security training for externals		—	—	—	—

Process

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
1	Definition of simple BCDR practices for critical components	A *Business Continuity and Disaster Recovery*(BCDR) is a plan and a process that helps a business to return to normal operations if a disaster…	✓	✓	✓	✓
2	Determining the protection requirement		—	—	—	—
3	Approval by reviewing any new version		—	—	—	—
3	Definition of a change management process		—	—	—	—

Implementation

Application Hardening

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
1	App. Hardening Level 1 (50%)	To tackle the security of code developed in-house, OWASP offers an extensive collection of Cheatsheetsdemonstrating how to implement features securely. Moreover, the Security Knowledge…	—	—	—	—
1	Context-aware output encoding	**Input validation** stops malicious data from entering your system. **Output encoding** neutralizes malicious data before rendering to user, or the next system. Input validation and…	✓	✓	✓	✓
1	Parametrization	By concatenating strings from user input to build SQL queries, an attacker can manipulate the query to do other unintentional SQL commands as well. This…	✓	✓	✓	✓
2	App. Hardening Level 1	To tackle the security of code developed in-house, OWASP offers an extensive collection of Cheatsheetsdemonstrating how to implement features securely. Moreover, the Security Knowledge…	—	—	—	—
2	Containers are running as non-root		—	—	—	—
3	App. Hardening Level 2 (75%)		—	—	◖	—
3	Secure headers		—	—	—	—
4	App. Hardening Level 2		—	—	—	—

Development and Source Control

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
1	Versioning	Use a version control system platform like Github, Gitlab, Bitbucket, gittea,... to version your code.	✓	✓	✓	✓
2	Require a PR before merging		◖	◖	—	—
3	Block force pushes		—	—	—	—
3	Dismiss stale PR approvals		—	—	—	—
3	Require status checks to pass		—	—	—	—
4	.gitignore		—	—	—	—

Infrastructure Hardening

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
1	MFA for admins		✓	✓	✓	✓
1	Simple access control for systems		—	—	—	—
1	Usage of edge encryption at transit		—	—	—	—
2	Applications are running in virtualized environments		—	—	—	—
2	Backup		—	—	—	—
2	Baseline Hardening of the environment		—	—	—	—
2	Isolated networks for virtual environments		—	—	—	—
2	MFA		—	—	—	—
2	Usage of an security account		—	—	—	—
2	Usage of encryption at rest		—	—	—	—
2	Usage of test and production environments		—	—	—	—
2	Virtual environments are limited		—	—	—	—
3	Filter outgoing traffic		—	—	—	—
3	Immutable infrastructure		—	—	—	—
3	Infrastructure as Code		—	—	—	—
3	Limitation of system events		—	—	—	—
3	Role based authentication and authorization		—	—	—	—
3	Usage of internal encryption at transit		—	—	—	—
3	Usage of security by default for components		—	—	—	—
3	WAF baseline	A baseline WAF configuration provides essential defense against common vulnerabilities, acting as a first line of automated threat detection and response. Steps: • Configure WAF in…	—	—	—	—
4	Hardening of the Environment		—	—	—	—
4	Production near environments are used by developers		—	—	—	—
4	Usage of a chaos technology		—	—	—	—
4	WAF medium	A medium-level WAF configuration builds upon the baseline to offer a more nuanced and responsive defense mechanism against a wider array of threats. Sample steps:	—	—	—	—

Information Gathering

Logging

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
1	Centralized system logging		—	—	—	—
2	Centralized application logging		—	—	—	—
2	Logging of security events	Implement logging of security relevant events. The following events tend to be security relevant: • successful/failed login/logout • creation, change, and deletion of users • errors during input…	—	—	—	—
3	Analyze logs		—	—	—	—
3	Visualized logging		—	—	—	—

Monitoring

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
1	Simple application metrics		—	—	—	—
1	Simple budget metrics		—	—	—	—
1	Simple system metrics		—	—	—	—
2	Alerting		—	—	—	—
2	Monitoring of costs		—	—	—	—
2	Visualized metrics		—	—	—	—
3	Advanced availability and stability metrics		—	—	—	—
3	Audit of system events		—	—	—	—
3	Deactivation of unused metrics		—	—	—	—
3	Grouping of metrics		—	—	—	—
3	Targeted alerting		—	—	—	—
4	Advanced app. metrics		—	—	—	—
4	Coverage and control metrics		—	—	—	—
4	Defense metrics		—	—	—	—
4	Screens with metric visualization		—	—	—	—

Test KPI

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
2	Number of vulnerabilities/severity	Communication can be performed in a simple way, e.g. text based during the build process. This activity depends on at least one security testing implementation.	—	—	—	—
2	Number of vulnerabilities/severity/layer	Communication can be performed in a simple way, e.g. text based during the build process. This activity depends on at least one security testing implementation....	—	—	—	—
2	Patching mean time to resolution via PR		—	—	—	—
3	Fix rate per repo/product		—	—	—	—
3	Generation of response statistics		—	—	—	—
3	SLA per criticality		—	—	—	—
4	Patching mean time to resolution via production		—	—	—	—

Test and Verification

Application tests

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
2	Security unit tests for important components		—	—	—	—
3	Security integration tests for important components		—	—	—	—
4	Smoke Test		—	—	—	—

Consolidation

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
1	Simple false positive treatment		—	—	—	—
1	Treatment of defects with severity high or higher		—	—	—	—
2	Artifact-based false positive treatment	Artifact-based false positive treatment enables more granular control over finding suppression by linking decisions to specific code artifacts, container images, or application versions. This approach…	—	—	—	—
2	Simple visualization of defects		—	—	—	—
3	Fix based on accessibility		—	—	—	—
3	Global false positive treatment	Global false positive treatment allows (security) teams to make organization-wide decisions about specific vulnerabilities or finding patterns. When a finding is marked as a false…	—	—	—	—
3	Integration in development process	Validating Findings by Security Engineers Pros: • Ensures accuracy and relevance of findings before they reach product teams • Reduces false positives, saving development teams time and…	—	—	—	—
3	Integration of vulnerability issues into the development process		—	—	—	—
3	Treatment of defects per protection requirement	The protection requirements for an application should consider: • Data criticality • Application accessibility (internal vs. external) • Regulatory compliance • Other relevant factors	—	—	—	—
3	Treatment of defects with severity middle		—	—	—	—
3	Usage of a vulnerability management system	For known vulnerabilities a processes to estimate the exploit ability of a vulnerability is recommended. To implement a security culture including training, office hours and…	—	—	—	—
4	Advanced visualization of defects		—	—	—	—
4	Reproducible defect tickets		—	—	—	—

Dynamic depth for applications

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
2	Coverage of client side dynamic components		—	—	—	—
2	Simple Scan		—	—	—	—
2	Usage of different roles		—	—	—	—
3	Coverage of hidden endpoints		—	—	—	—
3	Coverage of more input vectors		—	—	—	—
3	Coverage of sequential operations		—	—	—	—
4	Usage of multiple scanners		—	—	—	—

Dynamic depth for infrastructure

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
2	Test for exposed services		—	—	—	—
2	Test network segmentation		—	—	—	—
2	Test of the configuration of cloud environments		—	—	—	—
3	Test for unauthorized installation		—	—	—	—
3	Weak password test		—	—	—	—
4	Load tests		—	—	—	—

Static depth for applications

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
2	Software Composition Analysis (server side)	Use a tool like trivy and concentrate on application related vulnerabilities. At this stage, ignore vulnerabilities in container base images used in the service.	—	—	—	—
2	Test for Time to Patch		—	—	—	—
2	Test libyear		—	—	—	—
3	API design validation		—	—	—	—
3	Exploit likelihood estimation	Severity-based vulnerability triage alone generates a lot false positives, requiring a more refined approach.	—	—	—	—
3	Local development security checks performed		✓	—	◖	—
3	Software Composition Analysis (client side)		—	—	—	—
3	Static analysis for important client side components		—	—	—	—
3	Static analysis for important server side components		—	—	—	—
3	Test for Patch Deployment Time		—	—	—	—
4	Static analysis for all self written components		—	—	—	—
4	Usage of multiple analyzers		—	—	—	—

Static depth for infrastructure

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
1	Test for stored secrets in build artifacts		✓	✓	✓	◖
1	Test for stored secrets in code		—	—	—	—
2	Test cluster deployment resources		—	—	—	—
2	Test for image lifetime		—	—	—	—
2	Test of virtualized environments		—	—	—	—
2	Test the cloud configuration		—	—	—	—
2	Test the definition of virtualized environments		—	—	—	—
3	Test for malware		—	—	—	—
3	Test for new image version		—	—	—	—
4	Correlate known vulnerabilities in infrastructure with new image versions		—	—	—	—
4	Software Composition Analysis	Subscribing to Github projects and reading release notes might help. Software Composition Analysis for infrastructure might help, but is often too fine-granular.	—	—	—	—
4	Test of infrastructure components for known vulnerabilities		—	—	—	—

Test-Intensity

Level	Activity	Description	Team-alpha	Team-beta	Team C	Team D
1	Default settings for intensity		✓	✓	✓	✓
2	Regular automated tests		—	—	◖	—
3	Deactivating of unneeded tests		—	—	—	—
3	High test intensity		—	—	—	—
4	Creation and application of a testing concept		—	—	—	—