CYBERSECURITY AND SECURITY ARCHITECTURE FIELD GUIDE

Developing and Manufacturing Medical Devices

A practical guide for Cybersecurity Engineers and Security Architects

From security architecture and threat modelling to secure release, vulnerability management and resilient post-market operation

General cross-market guidance | Edition 1.0 | August 2026

Table of Contents

How to use this guide

This guide explains how cybersecurity engineers and security architects contribute to developing, manufacturing, releasing and maintaining medical devices. It covers connected hardware, embedded software, applications, cloud services, interoperability, manufacturing systems and supporting infrastructure where compromise could affect the device, its data or its safe operation.

CORE MESSAGE Medical-device cybersecurity protects safety, effectiveness, data and continued operation throughout the supported life. It must be engineered into the product and its ecosystem, evidenced under the quality management system and maintained after release.

Scope and important limitations

- Apply the organisation's approved cybersecurity, software-lifecycle, risk-management, design-control, privacy and quality procedures.
- Distinguish product cybersecurity from enterprise security while controlling the interfaces between them.
- Scale rigour to patient impact, exposure, exploitability, complexity, novelty and the supported operational environment.
- Do not make intended-purpose, regulatory-classification, clinical or residual-risk decisions in isolation.
- Confirm current standards, recognised editions and market-specific expectations through authorised Quality and Regulatory specialists.

1. The cybersecurity engineering and architecture role

Cybersecurity engineers and security architects turn product, clinical, safety and regulatory needs into enforceable technical controls. Their work creates the structure and evidence needed to prevent, detect, withstand, recover from and learn from security events.

Understand the contribution

Contribution	Practical outcome
Architecture	Trust boundaries, attack surfaces, security domains and failure containment are explicit.
Engineering	Security requirements and controls are implemented, reviewed and verified.
Assurance	Evidence connects threats, risks, controls, tests, anomalies and the released configuration.
Lifecycle support	Vulnerabilities, incidents, updates and obsolescence are managed after release.

Preserve cross-functional ownership

Security specialists provide challenge and technical judgement, while the legal manufacturer retains responsibility for product safety, quality, regulatory compliance and authorised release.

2. Establish the product and cybersecurity boundary

Define what belongs to the medical device and what belongs to its operating ecosystem. Cybersecurity scope normally extends beyond the marketed software binary to interfaces, update systems, credentials, manufacturing services and dependencies.

Map the complete ecosystem

- Device hardware, firmware, boot chain, service interfaces and removable media.
- Mobile, desktop and web applications used with the device.
- Cloud services, APIs, data stores, identity services and administrative portals.
- Hospital, laboratory, home, network and customer-controlled components.
- Manufacturing, provisioning, signing, update and support systems.
- Third-party services, libraries, platforms and supply-chain dependencies.

Record responsibility and assumptions

For each boundary, identify the owner, data exchanged, trust basis, security controls, monitoring, update responsibility and behaviour when an assumption fails.

3. Regulatory and quality-system foundations

Product cybersecurity evidence belongs within design controls and the quality management system. Applicable obligations depend on intended markets, device type, connectivity, claims and the consequences of compromise.

Know the principal references

- IEC 81001-5-1 for security activities across the health-software product lifecycle.
- IEC 62304 for medical-device software lifecycle processes.
- ISO 14971 and ISO/TR 24971 for safety risk management.
- ISO 13485 for controlled design, suppliers, records, CAPA and release.
- Current FDA premarket and post-market cybersecurity expectations, including statutory requirements where applicable.
- EU MDCG cybersecurity guidance and other market-specific requirements.

Use current controlled interpretations

Standards and guidance should be translated into project plans, deliverables and acceptance criteria through approved procedures. Record applicability, edition, rationale and any justified tailoring.

4. Cybersecurity planning and governance

The cybersecurity plan should establish scope, responsibilities, methods, deliverables, reviews and lifecycle interfaces. It should be integrated with development, risk, clinical, privacy, supplier and post-market plans.

Define an executable plan

- Products, releases, software items, environments and lifecycle phases in scope.
- Roles, competence, independence, decision authority and escalation paths.
- Threat modelling, risk assessment, architecture, implementation and verification activities.
- SBOM, vulnerability, incident, update, disclosure and communication processes.
- Supplier evidence, tool assurance, records, repositories and retention.
- Milestones, entry criteria, completion criteria, reviews and approvals.

Maintain the plan as the system changes

Update the plan when intended use, architecture, attack surface, suppliers, regulatory strategy or support model changes. A static template is not an effective control.

5. Security objectives and safety integration

Confidentiality, integrity and availability remain important, but medical devices also require authenticity, accountability, resilience and safe behaviour under attack. Security failures must be analysed for their effect on patients and users.

Connect security to product outcomes

- Integrity and authenticity of measurements, commands, software and clinical output.
- Availability of functions needed for timely diagnosis, treatment or monitoring.
- Confidentiality of personal, clinical, proprietary and credential data.
- Accountability for security-relevant and regulated actions.
- Resilience, containment, recovery and safe degradation.
- Protection of risk controls, essential performance and evidence needed for investigation.

Coordinate safety and security risk

A security scenario can initiate or defeat a safety control, while a security control can introduce a new use or availability hazard. Maintain explicit links and resolve conflicts jointly.

6. Asset identification and security context

Threat modelling begins with an accurate view of what must be protected and why. Assets include more than data; functions, credentials, software, configuration, time, evidence and update authority may be safety-critical.

Build a useful asset inventory

- Clinical data, patient identifiers, measurements, results and treatment parameters.
- Software, firmware, models, configuration, calibration and reference data.
- Cryptographic keys, certificates, secrets, tokens and recovery material.
- Safety functions, alarms, command paths and operational availability.
- Audit records, logs, timestamps, provenance and regulatory evidence.

- Build, signing, deployment, service and vulnerability-management capabilities.

Assign security properties and owners

For each asset, define required protection, owner, location, lifecycle, authorised actors and the consequence of loss, alteration, disclosure or unavailability.

7. Security architecture views and trust boundaries

Architecture should show how the product enforces trust, separates domains and contains compromise. Several complementary views are usually needed because a single network diagram cannot communicate all security decisions.

Maintain the essential views

- System context with users, external entities and deployment environments.
- Logical architecture with security domains, services and responsibilities.
- Data-flow views with trust boundaries, protocols, stores and transformations.
- Identity, key, update and administrative-control architectures.
- Security use cases and sequences for authentication, update, service and recovery.
- Deployment and manufacturing views showing provisioning and operational ownership.

Record architecture decisions

Document the decision, alternatives, assumptions, security rationale, risk impact and verification approach. Review decisions when the system or threat environment changes.

8. Threat modelling

Threat modelling systematically identifies how actors could misuse interfaces, data flows, privileges and dependencies. It should begin early and continue through implementation, change and post-market learning.

Use a repeatable method

- Define scope, assumptions, assets, actors, entry points and trust boundaries.
- Decompose functions, data flows, stores, protocols and dependencies.
- Apply an appropriate method such as STRIDE, attack trees or misuse cases.
- Consider local, remote, supply-chain, insider and physical attackers.

- Record scenarios, preconditions, affected assets, consequences and existing controls.
- Review completeness with system, software, hardware, clinical and operational specialists.

Avoid checklist-only modelling

A threat category list helps prompt analysis but does not replace understanding how an attacker moves through the actual architecture and affects product outcomes.

9. Cybersecurity risk assessment and acceptability

Cybersecurity risk assessment evaluates threat scenarios using an approved method that accounts for exploitability, exposure, impact and relevant safety consequences. It should support prioritisation without pretending that uncertain values are exact.

Make the assessment traceable

- Link the scenario to assets, architecture, vulnerabilities and possible harms.
- Describe attacker capability, access, prerequisites and plausible attack path.
- Assess clinical, privacy, operational, business and regulatory consequences.
- Identify controls, residual exposure, assumptions and evidence.
- Apply approved acceptability criteria and document decision authority.
- Reassess after design change, new vulnerability, incident or threat intelligence.

Separate related decisions

Cybersecurity risk rating, safety residual-risk acceptability, privacy impact and regulatory reportability are related but distinct. Keep their methods and approvals clear.

10. Security requirements and traceability

Security requirements should state enforceable, verifiable behaviour. They should derive from architecture, threats, risks, standards, privacy obligations and operational needs rather than generic control catalogues alone.

Write complete requirements

- Identify actor, asset, condition, required behaviour and measurable acceptance criterion.
- Specify normal, denied, fault, degraded, recovery and maintenance behaviour.

- Define algorithms, protocol properties, key lengths or approved policy references where needed.
- Address secure defaults, configuration limits, logging and failure response.
- Allocate each requirement to identifiable components and interfaces.
- Maintain links to source, design, implementation, verification and anomalies.

Avoid non-testable language

Terms such as secure, strong, industry standard and best practice are not sufficient requirements. State the concrete protection or controlled reference that determines acceptance.

11. Defence in depth and least privilege

Defence in depth uses independent or complementary controls so that one weakness does not expose the full product. Least privilege limits the authority available to users, devices, services and support personnel.

Design layered protection

- Minimise exposed services, ports, commands, privileges and stored secrets.
- Separate safety functions, administrative interfaces and externally reachable components.
- Use authenticated interfaces, validation, authorisation and integrity checks at boundaries.
- Limit service accounts, processes, containers and devices to required capabilities.
- Detect control failure and constrain lateral movement and persistence.
- Provide recoverable, auditable emergency and service access.

Demonstrate independence

Two controls implemented by the same component, credential or assumption may fail together. Analyse common cause and the actual independence of claimed layers.

12. Identity, authentication and authorisation

Identity architecture should cover humans, devices, applications, services and manufacturers across enrolment, use, recovery, transfer and retirement.

Control the complete identity lifecycle

- Unique identities and secure binding to the relevant user, device or service.
- Proportionate authentication, including stronger protection for privileged actions.
- Role- or attribute-based authorisation with deny-by-default behaviour.
- Controlled enrolment, approval, credential issuance, renewal and revocation.
- Secure recovery, break-glass access and separation of duties.
- Auditable changes to roles, policies, credentials and privileges.

Test failure and abuse paths

Verify expired, revoked, duplicated, lost, stolen and incorrectly provisioned credentials, as well as brute-force, enumeration, replay and privilege-escalation attempts.

13. Cryptography and key management

Cryptography should protect data and commands in transit and at rest, and where necessary during processing. Its effectiveness depends on algorithms, protocols, implementation and the complete key lifecycle.

Build a managed cryptographic system

- Use approved, current algorithms and protocols appropriate to product life and markets.
- Generate keys with sufficient entropy and protect private or secret material.
- Define ownership, storage, access, rotation, renewal, backup and destruction.
- Validate certificates, peers, purposes, chains, expiry and revocation where applicable.
- Plan crypto-agility for algorithm, certificate and trust-anchor changes.
- Handle clock failure, unavailable revocation services and recovery securely.

Avoid proprietary cryptography

Custom algorithms and unaudited protocols are difficult to assess and maintain. Prefer established constructions and vetted implementations with controlled configuration.

14. Secure boot, software integrity and device identity

A connected device should establish that it is running authorised software and presenting an authentic identity before trusted interaction occurs.

Protect the trust chain

- Anchor trust in protected hardware or an appropriately secured root.
- Verify boot stages, firmware, configuration and critical data before use.
- Prevent unauthorised rollback where older versions create unacceptable exposure.
- Protect debug, test and manufacturing modes from uncontrolled field use.
- Provision unique device identities and credentials under controlled conditions.
- Detect integrity failure and enter a specified recoverable or safe state.

Plan serviceability

Security controls should allow authorised diagnosis and repair without universal credentials or uncontrolled bypasses. Service paths require explicit risk analysis and auditability.

15. Secure communications and interoperability

Security properties must survive real protocols, gateways, proxies, message brokers and clinical interoperability workflows. Protect semantics and transaction state, not only the network channel.

Control communications end to end

- Authenticate communicating parties and authorise the requested operation.
- Protect confidentiality, integrity, freshness and replay resistance as required.
- Validate schema, length, range, identifiers, units, ordering and state.
- Define timeout, retry, duplication, partial success and recovery behaviour.
- Version interfaces and control downgrade, negotiation and deprecation.
- Test malformed, adversarial, delayed and out-of-sequence messages.

Recognise terminated trust

Encryption may terminate at a gateway, load balancer or integration engine. Document where data becomes visible and how trust and integrity are preserved beyond that point.

16. Data protection, privacy and auditability

Medical-device security must protect data through collection, use, sharing, support, retention and deletion. Logs and audit trails are valuable only when reliable and appropriately minimised.

Engineer governed data handling

- Classify data and document authorised purposes, users, locations and transfers.
- Minimise collection, replication, logging, export and retention.
- Protect stored data, backups, caches, removable media and support packages.
- Preserve provenance, correction history and integrity of regulated records.
- Restrict and monitor privileged data access and administrative exports.
- Apply approved deletion, anonymisation, retention and legal-hold rules.

Balance observability and exposure

Do not place secrets, credentials or unnecessary health data in telemetry. Record enough context to investigate safety and security events without creating an uncontrolled data store.

17. Secure software development practices

Secure development combines prevention, detection and evidence throughout requirements, design, implementation and maintenance. It should be part of normal engineering rather than a final review gate.

Embed security in development

- Approved languages, frameworks, coding rules and compiler protections.
- Peer review focused on trust boundaries, input handling, privileges and error paths.
- Static analysis, secret scanning, dependency analysis and appropriate dynamic testing.
- Safe memory, concurrency, parsing, serialisation and error-handling practices.
- Controlled generated code, scripts, build flags and environment variables.
- Security-focused unit, integration and regression tests linked to requirements.

Address findings at the source

Correct the requirement, design or process weakness that produced a defect. A local patch without systemic learning can allow the same vulnerability pattern to recur.

18. SOUP, open-source and software supply chain

Third-party software, services and build inputs form part of the attack surface. The manufacturer needs enough visibility and control to assess, reproduce, update and support the released product.

Govern components and suppliers

- Record identity, version, source, licence, intended use and transitive dependencies.
- Evaluate maintenance, known anomalies, vulnerabilities, provenance and alternatives.
- Pin, verify and protect packages, container images, toolchains and build inputs.
- Control repositories, registries, mirrors, signing and supplier access.
- Assess cloud services, operating systems and externally maintained components.
- Monitor change, deprecation and end-of-support throughout product life.

Treat provenance as evidence

A familiar package name is not proof that the correct artefact was obtained. Preserve hashes, sources, signatures and build provenance for the released configuration.

19. SBOM, VEX and dependency intelligence

A software bill of materials supports vulnerability identification and lifecycle decisions. VEX communicates whether a known vulnerability affects a particular product configuration and why.

Create actionable component information

- Use consistent component identifiers, versions, suppliers, hashes and relationships.
- Include direct, transitive, embedded, container and relevant service dependencies.
- Link each SBOM to the exact product release and deployment configuration.
- Reconcile generated output with architecture and build records.
- Document affected, not affected, fixed or under-investigation VEX status with rationale.
- Protect sensitive detail while meeting regulatory and customer obligations.

Do not overclaim completeness

Automated tools can omit statically linked, generated, vendored or runtime components. Define scope and known limitations, then use several evidence sources to improve coverage.

20. Security verification and penetration testing

Security verification should demonstrate that specified controls work and that plausible attack paths have been challenged. Penetration testing complements, but does not replace, requirement-based verification.

Build layered assurance

- Architecture and threat-model review by competent, proportionately independent reviewers.
- Requirement-based positive, negative, boundary and abuse-case tests.
- Static, dynamic, composition, interface and configuration analysis.
- Authentication, authorisation, cryptographic and update-path verification.
- Fuzzing, malformed input, protocol, resilience and fault-injection testing.
- Penetration testing in a representative integrated configuration.

Control test scope and findings

Identify build, environment, credentials, tools, rules of engagement, limitations and affected interfaces. Trace findings to risk, correction, retest, regression and approved disposition.

21. Secure update and patch architecture

Update capability is a central lifecycle control. It must deliver authentic, compatible software without creating unacceptable interruption, rollback, recovery or supply-chain risk.

Design a trustworthy update path

- Authenticate update authority and verify package integrity and intended target.
- Protect signing keys, release metadata, distribution and administrative access.
- Check version, hardware, dependencies, capacity, power and preconditions.
- Use atomic or recoverable installation with defined failure behaviour.

- Control rollback, anti-rollback exceptions and emergency recovery.
- Preserve audit evidence and confirm the installed configuration after update.

Plan for the field reality

Consider intermittent connectivity, low power, missed updates, unsupported intermediates, customer approval, fleet staging and devices that remain offline for long periods.

22. Manufacturing, provisioning and service security

Manufacturing and service processes can introduce credentials, enable interfaces and establish the product root of trust. Weak controls here can undermine an otherwise strong design.

Secure lifecycle transitions

- Protect golden images, test software, configuration and production programming.
- Provision unique identities, keys and certificates with auditable custody.
- Separate development, factory, service and production credentials and roles.
- Disable or control debug, calibration and test capabilities before release.
- Verify security configuration, secure boot and identity as production outputs.
- Control return, refurbishment, ownership transfer, data removal and disposal.

Avoid shared secrets at scale

Universal passwords, keys or service bypasses create systemic exposure. Where legacy constraints exist, document compensating controls and a remediation strategy.

23. Cloud, mobile and application security architecture

Applications and cloud services extend the device trust model into platforms partly controlled by suppliers, customers and users. Security architecture should reflect shared responsibility and rapid platform change.

Control the distributed product

- Tenant isolation, identity federation, administrative planes and privileged support.
- API gateways, service identities, secrets, network policy and workload isolation.
- Mobile storage, operating-system permissions, app signing and store distribution.

- Cloud regions, backups, managed services, subprocessors and supplier access.
- Infrastructure as code, configuration drift, deployment provenance and rollback.
- Monitoring for abuse, dependency failure and unauthorised configuration change.

Retain manufacturer responsibility

Supplier certifications and shared-responsibility models support assurance but do not prove that the configured medical-device service is secure or compliant.

24. Security monitoring and threat intelligence

Post-market security depends on timely information about product behaviour, supplier changes, emerging threats and vulnerabilities. Monitoring should be proportionate, privacy-aware and linked to action.

Build an actionable monitoring system

- Product telemetry, security events, complaints, service data and anomaly trends.
- Vulnerability feeds, supplier notices, researcher reports and sector intelligence.
- Asset, version, deployment and customer information needed to determine exposure.
- Alert ownership, thresholds, triage, escalation and response times.
- Detection of missing telemetry, disabled controls and monitoring gaps.
- Periodic review of effectiveness, false positives and undetected events.

Connect intelligence to the product

A vulnerability headline is not an exposure assessment. Determine the affected component, version, configuration, attack path, compensating controls and possible safety consequences.

25. Vulnerability management and coordinated disclosure

Vulnerability management should provide a trusted route to receive, assess, remediate and communicate security findings throughout the supported life.

Operate a controlled process

- Publish monitored researcher contact and coordinated vulnerability disclosure information.

- Acknowledge reports, protect sensitive information and avoid unnecessary researcher friction.
- Reproduce and assess affected products, versions, exploitability and patient impact.
- Link technical analysis to safety risk, privacy, regulatory and reporting decisions.
- Develop, verify, approve and distribute remediation or compensating measures.
- Communicate status, timelines, residual risk and customer actions appropriately.

Measure more than closure speed

Track ageing, recurrence, affected fleet, time to containment, remediation adoption and control effectiveness. Rapid closure without robust assessment can conceal unresolved exposure.

26. Incident response, recovery and regulatory interfaces

A product-security incident may also be a safety event, complaint, privacy breach, quality nonconformity or reportable regulatory event. Response roles and evidence paths should be defined before an incident occurs.

Prepare coordinated response

- Detection, triage, command structure, decision authority and out-of-hours escalation.
- Safe containment that preserves clinical continuity and forensic evidence.
- Affected product, fleet, data, customer, geography and time-period assessment.
- Links to complaint handling, vigilance, privacy, CAPA and regulatory reporting.
- Verified recovery, remediation, customer communication and effectiveness monitoring.
- Exercises covering technical, clinical, operational and executive decisions.

Preserve evidence carefully

Collect logs, volatile data, versions, configuration, time sources and actions under controlled custody. Avoid destructive troubleshooting before essential evidence is secured.

27. Cybersecurity release evidence and residual risk

Release evidence should present a balanced, configuration-specific security case. It should show what was planned, implemented, verified, unresolved and accepted by authorised roles.

Assemble the release baseline

- Approved cybersecurity plan, scope, architecture, threat model and risk assessment.
- Traceable security requirements, design decisions and implemented controls.
- Verification, penetration-test, review and anomaly-resolution evidence.
- SBOM, VEX status, known vulnerabilities and dependency assessment.
- Update, monitoring, disclosure, incident and support readiness.
- Residual risks, limitations, labelling, approvals and exact released configuration.

Avoid absolute claims

Statements such as secure or vulnerability-free are rarely defensible. Describe scope, methods, evidence, known limitations and the supported conditions of use.

28. Maintenance, legacy devices and retirement

Cybersecurity obligations continue while products remain supported and may persist during transition and retirement. Legacy constraints require explicit risk-based management rather than silent acceptance.

Manage the supported life

- Define support period, compatible platforms, update channels and customer responsibilities.
- Monitor vulnerabilities, threats, suppliers, cryptography and operational performance.
- Maintain build, test, signing, recovery and investigation capability.
- Prioritise remediation using exploitability, exposure and patient consequences.
- Communicate compensating controls, limitations, end of support and migration options.
- Retire credentials, services and data safely while retaining required records.

Plan secure retirement early

Architecture choices determine whether devices can be patched, migrated and decommissioned safely. Treat maintainability and crypto-agility as design inputs, not future assumptions.

A practical cybersecurity engineering sequence

Use this sequence for a new device, connected function, major architecture change or significant maintenance release. Adapt the depth to risk, exposure and the approved lifecycle.

1. **Establish the context.** Confirm intended use, product boundary, users, assets, deployment, regulatory scope and security objectives.
2. **Model the system and threats.** Create architecture and data-flow views, identify trust boundaries, attack paths, scenarios and safety consequences.
3. **Define and implement controls.** Translate risk into traceable requirements, architecture decisions, secure implementation and supplier controls.
4. **Verify the security case.** Review, analyse and test controls, attack paths, updates, resilience and the representative integrated configuration.
5. **Approve and release.** Reconcile findings, SBOM/VEX, residual risk, monitoring, response readiness and the exact release baseline.
6. **Monitor and maintain.** Assess intelligence, vulnerabilities and incidents; deploy verified remediation and manage support through retirement.

Appendix A. Cybersecurity deliverables by lifecycle stage

Stage	Typical controlled deliverables
Concept and scope	Cybersecurity boundary, ecosystem view, security objectives, preliminary assets and regulatory assumptions.
Planning	Cybersecurity plan, responsibilities, methods, supplier strategy, monitoring, disclosure and maintenance plans.
Requirements	Traceable security, privacy, safety-interface, update, resilience and operational requirements.
Architecture and design	Security views, trust boundaries, threat model, risk assessment, decisions and control allocation.

Stage	Typical controlled deliverables
Implementation	Reviewed code and configuration, component evidence, build provenance, SBOM and resolved findings.
Verification	Requirement tests, analyses, penetration testing, update and recovery evidence, anomalies and reports.
Release	Cybersecurity report, SBOM/VEX, known-vulnerability assessment, residual risk and approved baseline.
Post-market	Monitoring, vulnerability, incident, disclosure, remediation, support and retirement records.

Appendix B. Essential cybersecurity review checklist

Review area	Minimum question
Scope	Does the cybersecurity boundary include the complete product ecosystem and lifecycle dependencies?
Architecture	Are assets, trust boundaries, identities, data flows, privileges and failure containment explicit?
Threats	Do threat scenarios cover credible local, remote, supply-chain, insider and physical paths?
Risk	Are exploitability, exposure, consequences, safety links, controls and acceptability traceable?
Requirements	Are security controls objective, allocated, verifiable and linked to their source?
Implementation	Are code, configuration, cryptography, dependencies and build provenance controlled?
Verification	Have controls and plausible attack paths been challenged in a representative configuration?
Updates	Can the product authenticate, install, recover from and evidence updates safely?
Post-market	Can the manufacturer monitor, assess, disclose, remediate and communicate vulnerabilities?
Release	Are residual risk, SBOM/VEX, limitations, approvals and the exact baseline reconciled?

Appendix C. Commonly relevant standards and guidance

This orientation list is not exhaustive. Confirm applicability, current editions, recognition and product-specific interpretation through approved Quality and Regulatory processes.

Topic	Common reference	Cybersecurity relevance
Security lifecycle	IEC 81001-5-1:2021	Security activities across development, maintenance and post-market support.
Software lifecycle	IEC 62304:2006 + A1:2015	Controlled software development, maintenance and problem resolution.
Risk management	ISO 14971:2019; ISO/TR 24971:2020	Safety consequences, risk controls, verification and residual risk.
Quality management	ISO 13485:2016; US 21 CFR Part 820 QMSR	Design controls, suppliers, records, CAPA and release.
FDA cybersecurity	Current FDA medical-device cybersecurity guidance	Premarket documentation, secure lifecycle, SBOM and post-market capability.
EU cybersecurity	MDCG 2019-16 Rev.1	Cybersecurity expectations under EU medical-device regulations.
Secure development	NIST SP 800-218, Secure Software Development Framework	Organisation-level secure development practices and evidence.
Interoperability	Applicable device, health-data and protocol standards	Secure interface contracts, authentication, integrity and resilience.

Appendix D. Glossary

Term	Meaning in this guide
Asset	Data, function, capability, evidence or resource requiring protection.
Attack surface	The interfaces, functions and conditions through which an attacker may interact with the product.
Cybersecurity risk	Risk arising from threats exploiting vulnerabilities and affecting protected product outcomes.

Term	Meaning in this guide
Defence in depth	Layered controls intended to prevent one failure from exposing the complete system.
SBOM	A software bill of materials identifying software components and dependency relationships.
SOUP	Software of unknown provenance or without adequate development-process evidence for the manufacturer's needs.
Threat	A circumstance or event with the potential to adversely affect an asset.
Threat model	A structured representation of assets, actors, architecture, attack paths, scenarios and controls.
Trust boundary	A point at which data or control crosses between entities or domains with different trust assumptions.
VEX	Vulnerability Exploitability eXchange information describing the status and rationale of vulnerabilities for a product.
Vulnerability	A weakness that may be exploited or triggered to compromise a security objective.
Zero trust	An approach that avoids implicit trust based solely on network location and continuously verifies access.

Appendix E. Authoritative starting sources

- IEC — IEC 81001-5-1, Security activities in the product life cycle
- ISO — IEC 62304, Medical device software lifecycle processes
- ISO — ISO 13485:2016, Medical devices quality management systems
- ISO — ISO 14971:2019, Application of risk management to medical devices
- ISO — ISO/TR 24971:2020, Guidance on ISO 14971
- FDA — Cybersecurity in Medical Devices
- FDA — Medical Device Cybersecurity
- FDA — Quality Management System Regulation
- European Commission — MDCG 2019-16 Rev.1, Guidance on Cybersecurity for Medical Devices
- IMDRF — Principles and Practices for Medical Device Cybersecurity
- NIST — SP 800-218, Secure Software Development Framework
- CISA — Software Bill of Materials
- CycloneDX — Software Bill of Materials standard