CRA from regulation to reality: How to secure digital products

Content

Introduction

Why is this quick guide to cybersecurity best practices essential for software development?

In today's digital world, cybersecurity threats are growing not only in number but also in complexity. That is precisely why it is crucial to consider security aspects as an integral part of software development right from the start.

Recent reports show that the average cost of a data breach has now reached alarming levels – a clear signal of how important it is to integrate security measures throughout the entire lifecycle of an application. Regulatory developments such as the **EU Cyber Resilience Act (CRA)** also underscore this necessity by introducing mandatory security requirements for digital products and software.

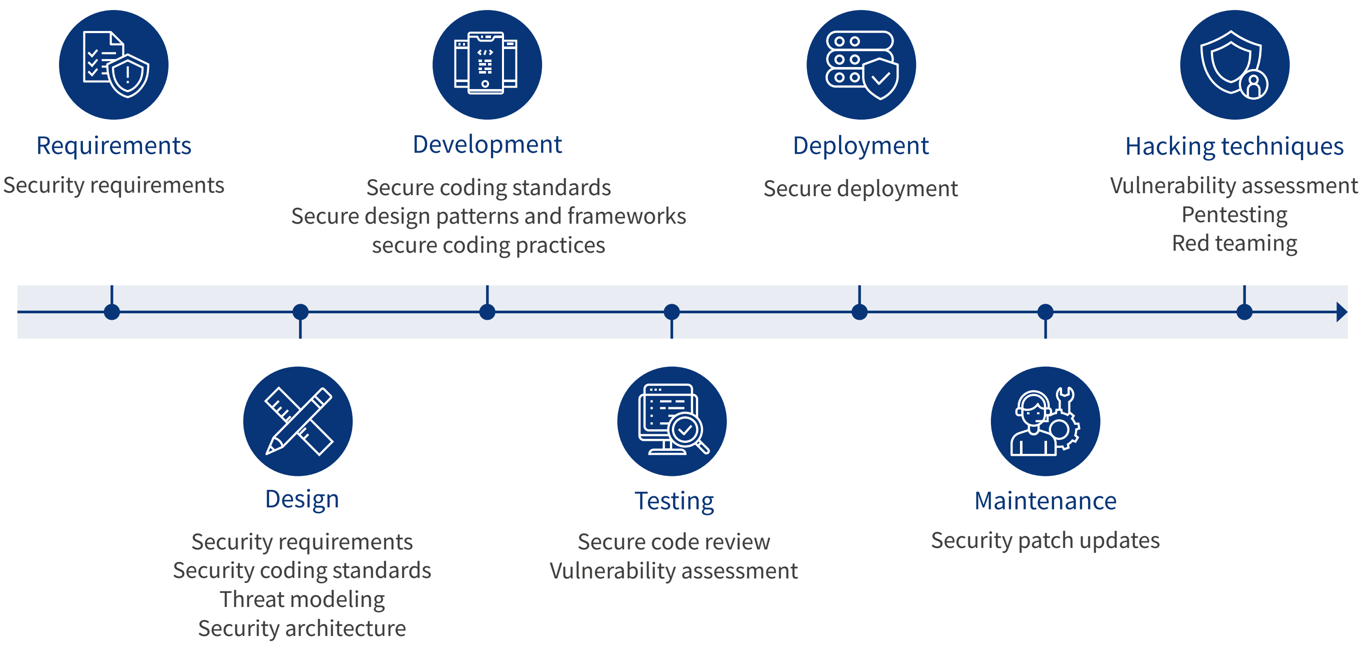
Every day, thousands of new applications and 'connected devices' enter the market – many of which quickly become targets for cybercriminals. Companies are responsible for developing software that is not only functionally powerful but also fundamentally secure.

In this eBook, we present best practices that can be applied throughout the entire lifecycle of digital solutions – from requirements analysis to updates and maintenance during operation. The goal is not only to reduce security risks but also to promote a sustainable culture of secure development.

Our approach is based on the principles of Swiss Software Engineering: we see the development of digital solutions as a process that is economically efficient and sustainable in the long term.

We place particular emphasis on quality, security, clear responsibility, and resilience in the face of technical and operational challenges. These principles are reflected in a practice-oriented implementation that is geared toward the regulatory, security-related and industry-specific requirements of the German market.

This results in solutions that not only impress with their functions but also withstand current threats and meet regulatory requirements. Proven standards such as OWASP, NIST and FAIR, as well as modern tools, are a great help in implementation.



Design and definition of requirements

Every major project starts with a solid foundation. In this initial phase, it is important to lay out the path in such a way that clear and precise requirements are defined, based on five fundamental pillars:

FIVE FUNDAMENTAL PILLARS

- **Threat modelling:** Depending on how complex and critical the system is, methods such as STRIDE (for quick, component-based threat analysis) or PASTA (risk-based, business-centric approach) should be used. This enables effective controls that prevent threats before they actually materialise.

- **Analysis of data flows:** Data flows within a system should be analysed and visualised at an early stage – ideally using data flow diagrams (DFD). Particular attention should be paid to trust boundaries, i.e., interfaces where data crosses different levels of trust. This form of structured modelling helps to identify potential attack points at an early stage and ensure that sensitive information is adequately protected, treated confidentially and protected from manipulation throughout its entire lifecycle.

- **Creation of clear requirements:** The security requirements of a project should be documented clearly and early on – ideally based on established references such as the OWASP Top 10 and the NIST guidelines. Such standards help to systematically address common threats and anchor security-related requirements as an integral part of the technical specification.

- **Cryptography and key management:** Protecting sensitive information requires that companies use secure encryption methods and implement structured management of access data and secrets. Instead of storing secrets directly in source code or configuration files, consider using centrally managed secrets management solutions that support access controls, logging and regular key rotation. Solutions such as Vault by HashiCorp or Infisical can be integrated into modern development and operational processes and enable secure and traceable handling of confidential configuration data.

- **Security architecture:** A holistic security concept should be based on the principle of defence in depth: The interaction of several protection mechanisms at different levels – through network segmentation, strict access controls, and zero-trust architectures – ensures that even if an attack attempt is successful, the entire system is not compromised. Each individual layer of protection acts as a barrier that slows down, restricts or completely prevents attacks. Especially in the context of various ERNI projects, such as in public administration, medical and laboratory technology, healthcare, industrial automation or finance and insurance, there are high requirements for security, traceability and long-term maintenance. In these areas, the traceable, compliant implementation of security measures is not only a question of technical excellence, but often a necessity. Security concepts must therefore not only ward off attacks but also ensure auditability, accountability and sustainable operability in accordance with both the Cyber Resilience Act (CRA) and ISO standards.

In the DACH region, legal requirements such as the (revised) Data Protection Act (revDSG/GDPR) and industry-specific security requirements in the financial, healthcare and industrial sectors are becoming increasingly important. Projects that take security aspects into account at an early stage not only create technological robustness but also meet key requirements for transparency, traceability and compliance. Integrating such requirements into architecture, development and operations increases legal certainty and strengthens the trust of customers, partners and regulatory authorities alike.

Development and testing

Once a solid foundation has been established, the requirements are translated into code. Each line should be written with the clear goal of incorporating security from the outset. The focus should be on the following aspects:

FOCUS ON

- **Patterns and frameworks for secure design:** Secure software design begins with the consistent implementation of established security principles such as least privilege, defence in depth and fail secure by default. The use of proven frameworks such as Spring Security in Java environments or ASP.NET Core Identity in the .NET domain supports consistent implementation of authentication and authorisation. In modern, cloud-native architectures, a service mesh such as Istio with features such as mTLS and fine-grained access control can increase security between distributed components. Token-based authentication methods such as JWT also offer flexible options, but require the correct handling of token lifetimes, signature validation, and revocation strategies. The selection of technologies should always be based on the protection needs of the application and the relevant security requirements.

- **Dependency management:** A complete inventory of all software components – known as a software bill of materials (SBOM) – is a central foundation for the security of modern applications. It enables transparent tracking of all libraries and dependencies and helps to identify known vulnerabilities in third-party software at an early stage. Continuous maintenance and testing of the SBOM ensures that no outdated or insecure components are unintentionally left as part of the application and compromise the security base.

- **Practices for secure code:** Compliance with established standards for secure programming – such as the CERT Secure Coding Standards and the ISO/IEC 27001 guidelines in the development context – forms an important basis for avoiding typical security vulnerabilities such as injection attacks, race conditions or insecure memory access. While the SEI CERT rules provide specific recommendations for secure and error-free coding, ISO/IEC 27001 defines overarching requirements for secure processes – including in software development. Supplemented by manual code reviews and automated static analyses, this creates a robust code base.

- **Protocols for secure communication:** All data transmissions should be secured by proven security protocols such as TLS or IPSec to reliably ensure the confidentiality and integrity of the information. Ideally, implementation should be carried out using well-maintained crypto libraries – such as OpenSSL or comparable alternatives – so that sensitive data remains protected even when transmitted over insecure networks.

- **Analysis of code vulnerabilities:** To enable companies to identify and remedy security gaps at an early stage, a combination of static (SAST) and dynamic (DAST) analysis should be used. While static methods examine the source code for vulnerabilities during development, dynamic tests simulate real attacks on running applications. Tools such as OWASP ZAP or Burp Suite help you uncover weaknesses in the behaviour of the application before they reach productive environments.

Provision and maintenance

Application security does not end with development; it extends to deployment and maintenance. Therefore, it is important to implement strategies that promote a robust production environment.

KEEP THESE STRATEGIES IN MIND

- **Automated provisioning:** This can be achieved by using integration and deployment pipelines that automatically integrate security checks at every stage – through static code analysis, secrets scanning or container vulnerability checks. This allows risks to be systematically minimised before rollout.

- **Patch and update management:** A clearly defined process for applying security updates and patches reduces the attack surface and protects the infrastructure from known vulnerabilities. Rollback mechanisms should be in place to enable a quick response in the event of errors.

- **Monitoring and logging:** Continuous monitoring and centralised logging enable early detection of suspicious activity. They form the basis for an effective response in the event of security incidents.

- **Infrastructure-as-Code:** Infrastructure-as-Code (IaC) enables systems to be set up consistently and transparently. Changes to the infrastructure are versioned, documented in a traceable manner and can be rolled out automatically.

- **Containerised environments:** Container technologies such as Docker ensure consistent execution environments and facilitate the separation of responsibilities. In combination with platforms such as Kubernetes, you can automatically apply and enforce security policies.

Hacking techniques and defence assessment

In order to develop effective and robust defence mechanisms, it is important to view your own application from the perspective of a potential attacker. You should therefore regularly use methods that simulate real attack scenarios and critically review the effectiveness of existing protective measures.

WE SUGGEST

- **Penetration testing and red teaming:** Planned security tests help to uncover technical vulnerabilities under realistic conditions. While classic penetration tests specifically examine individual points of attack, red teaming takes a more comprehensive approach: it simulates targeted attacks on systems, processes and even employee behaviour in order to assess the overall security situation. The goal is to identify gaps early on – before they can be exploited by real attackers.

- **Vulnerability analysis:** Regular vulnerability assessments allow known security gaps in applications, system components, or configurations to be systematically identified. Automated scans are used for this purpose, as are manual checks. The findings should be evaluated on a risk basis and remedied in accordance with established best practices.

- **Tabletop exercise:** In addition to technical tests, organisational processes should also be reviewed regularly. A tabletop exercise simulates security incidents in a controlled environment to test the responsiveness of teams, communication chains and escalation processes. Such exercises help to clarify responsibilities and reveal weaknesses in existing emergency plans.

Conclusion

Security should not be an afterthought in software development – it is a fundamental prerequisite for resilient, trustworthy systems. This eBook shows you in a practical way how security aspects can be systematically embedded in every phase of the development process: from requirements definition and design to implementation, operation and ongoing maintenance.

The approaches described help to identify risks early on, close security gaps and create resilient applications in the long term. Whether through secure architecture decisions, automated testing mechanisms or a structured incident response strategy, it is crucial that technical measures, organisational processes and security awareness within the team go hand in hand.

Contact

ERNI Schweiz AG
Löwenstrasse 11
8001 Zurich

Phone: +41 58 268 12 00
Email: info@erni.ch

www.betterask.erni
Zurich I Basel I Bern I Lausanne I Lucerne

better ask ERNI