36 Concepts in Software Risk Management

Concept	Description
Attack Surface Analysis	Identifying and assessing potential points of entry or vulnerabilities in a system that an attacker could exploit. Do not forget the APIs!
Automated Dependency Updates	Automatically updating software dependencies to maintain security and ensure compatibility. This is different than simple updates.
Automated Enforcement	Using automated systems to block non-compliant software dependencies. This is different than application whitelisting.
Change Control Tracking	A systematic process for managing and documenting changes to a project or system.
Checksum Validation	Verifying the integrity of data by comparing a computed checksum with a pre-existing one. I use SHA-2 family for sub resource integrity for webserver (due to speed) and SHA-3-512 for critical applications.
Code Signing	Using digital signatures to verify the authenticity and integrity of software code, preventing tampering by unauthorised parties. I use ECDSA (P-384/P-521) for traditional certificates and at https://kyber.club you can create ML-DSA digital signatures for free.
DevSecOps	Integrating security practices throughout the entire software development lifecycle. My personal view is that agile practices rarely put security first!
Drift Detection	Identifying unintended or unauthorised changes in a system's configuration over time. Note that debsums is not comprehensive.
Dynamic Application Security Testing (DAST)	Testing running applications for vulnerabilities by simulating external attacks, without access to source code. OWASP testing is a bare minimum.
End-of-Life Identification	Determining when a software component or technology will no longer be supported or receive updates. Some vendors provide paid Extended Support for a limited period of time. I would rather upgrade than waste my money.
Fork and Customise External Code	Creating a copy of external code to modify it for specific needs while tracking changes. Forking helps customisation and protects against malicious upstream changes but adds to the maintenance burden.
Fuzz Testing	Automated software testing technique that inputs invalid, unexpected, or random data to discover vulnerabilities and crashes.

Concept	Description
Hermetic Builds	Build processes that are isolated from external networks and rely only on explicitly defined inputs, ensuring reproducibility.
Input Validation and Sanitisation	Ensuring user-supplied data is properly checked and cleaned to prevent injection attacks (e.g., SQL, command, or XSS injection) and similar input-related vulnerabilities.
Inventory Tracking	Maintaining a comprehensive record of all software components and assets.
License Compliance Checks	Verifying adherence to the licensing terms of software components.
Lockfiles with Hashes	Files that record the exact versions and cryptographic hashes of dependencies to ensure reproducible builds and prevent tampering.
Multi-layer Scanning	Performing security scans at different stages of the software development lifecycle and across various layers of the software stack.
Namespace Reservation	Reserving specific names in software registries to prevent malicious actors from publishing packages with similar names.
Open-source Software (OSS) Supply Chain Security	Securing the process of creating, distributing, and consuming open-source software.
Patch Management	Systematically identifying, acquiring, testing, and applying updates to software to address vulnerabilities and improve functionality. Do not forget the low CVEs!!!
Private Vulnerability Reporting	A mechanism for individuals to report security vulnerabilities directly to developers before public disclosure.
Provenance Data	Information about the origin and history of a software component, including how it was built and by whom.
Rate Limiting	Implementing controls to restrict the number of requests or resource usage per user/session, mitigating denial-of-service risks from unrestricted resource consumption in APIs.
Reproducible Outputs	Ensuring that a build process consistently produces the same output every time it is run with the same inputs.
SDLC (Software Development Life Cycle) Transparency	Ensuring visibility and accountability throughout all stages of software development.
Secure By Design	Incorporating security principles and practices into the initial design and architecture of software systems.

Concept	Description
Shift Left Security	Shifting security considerations and practices to earlier stages of the software development lifecycle (corrected from "Security Left" for accuracy).
Software Bill of Materials (SBOM)	A formal, machine-readable inventory of all components, dependencies, and metadata in a software product.
Software Escrow	Safe custody of source code with a third party to be used in case of rupture of a contract (e.g. bankruptcy of the service provider)
Static Application Security Testing (SAST)	Analysing source code or binaries for vulnerabilities without executing the program.
Third-Party Risk Management	Assessing and mitigating risks associated with external vendors, suppliers, and partners in the software ecosystem.
Threat Intelligence	Collecting and analysing information about current and emerging threats to proactively defend against them.
Threat Modelling	Systematically identifying, prioritising, and mitigating potential threats and vulnerabilities in a system.
Vulnerability Prioritisation	Ranking vulnerabilities based on their severity, exploitability, and potential impact.
Zero Trust Architecture	A security model that requires continuous verification of users, devices, and applications, assuming no inherent trust.

Special note: The idea that using a memory-safe language like Rust automatically eliminates all weaknesses in C is misleading. While Rust's ownership model and borrow checker prevent common C vulnerabilities like buffer overflows and dangling pointers, they do not address logic errors, misuse of unsafe code, or vulnerabilities in C-based dependencies. Memory safety reduces but does not eliminate all security risks.

All views in this note and all errors/mistakes/omissions are solely mine. It is just my preferred list.

Santosh Pandit

London 30 July 2025