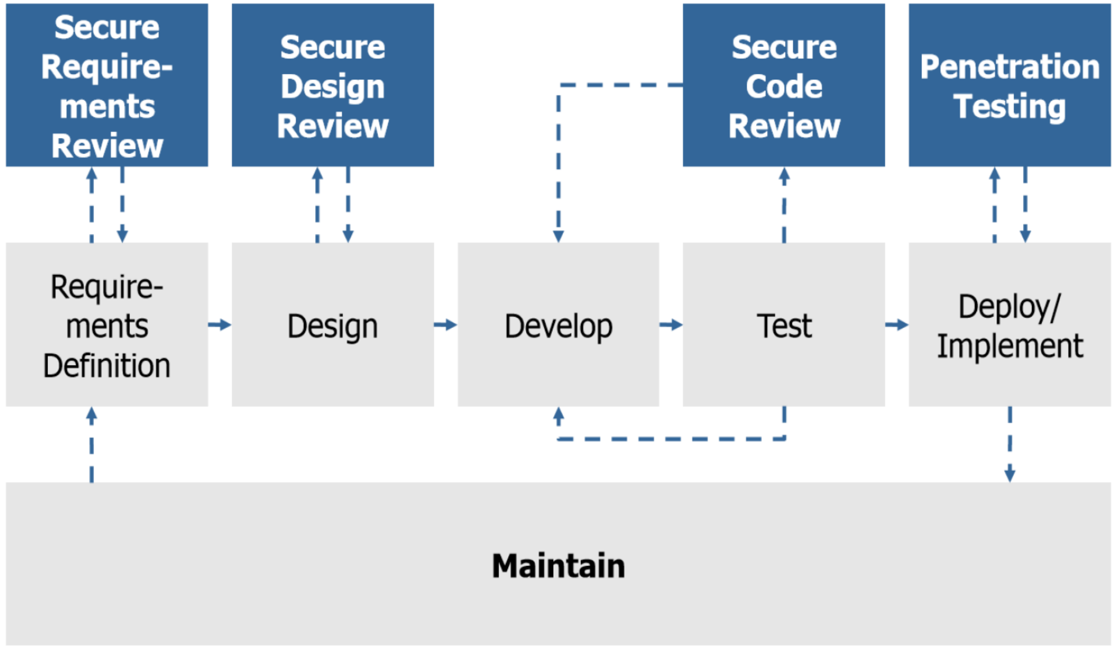
Развој безбедног софтвера

Методологије развоја безбедног софтвера

ISSES

Co-funded by the Erasmus+ Programme of the European Union

Сигурност у процесу развоја софтвера



ISSES
Co-funded by the Erasmus+ Programme of the European Union

Методологије за развој безбедног софтвера

- MS SDL (Microsoft Secure Development Lifecycle)
- OWASP CLASP (The Open Web Application Security Project Comprehensive, Lightweight Application Security Process)
- OSSA (Oracle Software Security Assurance)
- TSP-Secure (Team Software Process Secure)
- OWASP SAMM (Software Assurance Maturity Model)
- BSIMM (Building Security In Maturity Model Framework)
- NIST SSDF (Secure Software Development Framework)
- …

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Студија случаја: MS SDL

Свеобухватан процес посвећен производњи безбедног софтвера

ISSES

Co-funded by the Erasmus+ Programme of the European Union

Животни циклус сигурног развоја

Објављен од стране Microsoft-а пре више од 20 година

Пословни процес који помаже да се произведе безбеднији софтвер

Размотрићемо активности процеса, посебно:

- **Шта која активност подразумева**
- **Како се свака активност мапира на организацију**

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Core Security Training

Foundational concepts for building better software, including secure design, threat modeling, secure coding, security testing, and best practices surrounding privacy

ISSES

Co-funded by the Erasmus+ Programme of the European Union

Train	Require	Design	Develop	Verify	Release	Response

Establish Security and Privacy Requirements

Defining and integrating security and privacy requirements early helps to identify key milestones and deliverables and minimize disruptions to plans and schedules

Create Quality Gates/Bug Bars

Defining minimum acceptable levels of security and privacy quality at the start helps a team understand risks associated with security issues, identify and fix security bugs during development, and apply the standards throughout the entire project

Perform Security and Privacy Risk Assessments

Examining software design based on costs and regulatory requirements helps a team identify which portions of a project will require threat modeling and security design reviews and determine the Privacy Impact Rating of a feature, product, or service

ISSES Co-funded by the Erasmus+ Programme of the European Union

Train	Require	Design	Develop	Verify	Release	Response

Establish Design Requirements

Considering security and privacy concerns early helps minimize the risk of schedule disruptions and reduce a project's expense

Attack Surface Analysis/Reduction

Thoroughly analyzing overall attack surface and includes disabling or restricting access to system services, applying the principle of least privilege, and employing layered defenses

Use Threat Modeling

Applying a structured approach to threat scenarios during design helps a team identify vulnerabilities, determine risks, and establish appropriate mitigations

ISSES

Co-funded by the Erasmus+ Programme of the European Union

Train | Require | Design | Develop | Verify | Release | Response

Use Approved Tools

Publishing a list of approved tools and associated security checks (such as compiler/linker options and warnings) helps automate and enforce security practices easily at a low cost

Deprecate Unsafe Functions

Analyzing all project functions and APIs and banning those determined to be unsafe helps reduce potential security bugs

Perform Static Analysis

Analyzing the source code prior to compile provides a scalable method of security code review and helps ensure that secure coding policies are being followed

ISSES Co-funded by the Erasmus+ Programme of the European Union

Train	Require	Design	Develop	Verify	Release	Response

Perform Dynamic Analysis

Performing run-time verification checks software functionality using tools that monitor application behavior for memory corruption, user privilege issues, etc.

Fuzz Testing

Inducing program failure by deliberately introducing malformed or random data to an application helps reveal potential security issues prior to release

Attack Surface Review

Reviewing attack surface measurement upon code completion helps ensure that any design or implementation changes to an application or system have been taken into account, and that any new attack vectors have been reviewed and mitigated

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Train	Require	Design	Develop	Verify	Release	Response

Create an Incident Response Plan

Crucial for helping to address new threats that can emerge over time and includes identifying appropriate security emergency contacts and establishing security servicing plans for code inherited from other groups in or out of the organization

Conduct Final Security Review

Includes examining threat models, tools outputs, and performance against the quality gates and bug bars, to ensure software release readiness

Certify Release and Archive

Certifying software prior to a release helps ensure security and privacy requirements were met. Archiving all pertinent data is essential

ISSES

Co-funded by the Erasmus+ Programme of the European Union

Execute Incident Response Plan

Being able to implement the Incident Response Plan instituted in the Release phase is essential to helping protect customers from software security or privacy vulnerabilities that emerge

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Студија случаја: OWASP CLASP

Скуп активности које доводе до безбедног софтвера

ISSES

Co-funded by the Erasmus+ Programme of the European Union

Процес сигурног развоја

Објављен од стране OWASP-а пре више од 15 година

Скуп активности које помажу да се произведе безбеднији софтвер

Заснован на улогама (role-based), вођен активностима (activity-driven)

Размотрићемо активности, посебно:

- **Шта која активност подразумева**
- **Како се свака активност мапира на улогу у организацији**

ISSES

Co-funded by the Erasmus+ Programme of the European Union

Project Manager	Requirements Specifier	Architect	Designer	Implementer	Test Analyst	Security Auditor

Institute security awareness program

Ensure project members consider security to be an important project goal through training and accountability. Ensure project members have enough exposure to security to deal with it effectively.

Monitor security metrics

Gauge the likely security posture of the ongoing development effort. Enforce accountability for inadequate security.

Manage security issue disclosure process

Communicate effectively with outside security researchers when security issues are identified in released software, facilitating more effective prevention technologies. Communicate effectively with customers when security issues are identified in released software.

ISSES

Co-funded by the Erasmus+ Programme of the European Union

Project Manager	Requirements Specifier	Architect	Designer	Implementer	Test Analyst	Security Auditor

Specify operational environment

Document assumptions and requirements about the operating environment so that the impact on security can be assessed.

Identify global security policy

Provide default baseline product-security business requirements. Provide a means of comparing the security posture of different products across an organization.

Document security-relevant requirements

Document business-level and functional requirements for security.

Detail misuse cases

Communicate potential risks to stakeholder. Communicate rationale for security-relevant decisions to stakeholder.

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Project Manager	Requirements Specifier	Architect	Designer	Implementer	Test Analyst	Security Auditor

Identify resources and trust boundaries

Provide a structured foundation for understanding the security requirements of a system.

Identify user roles and resource capabilities

Define system roles and the capabilities/resources that the role can access.

Integrate security analysis into source management process (Integrator)

Automate implementation-level security analysis and metrics collection.

Perform code signing (Integrator)

Provide the stakeholder with a means of validating the origin and integrity of the software.

Project Manager	Requirements Specifier	Architect	Designer	Implementer	Test Analyst	Security Auditor

Identify attack surface

Specify all entry points to a program in a structured way to facilitate analysis.

Apply security principles to design

Harden application design by applying security design principles. Determine implementation strategies for security services. Design secure protocols and APIs.

Research and assess security posture of technology solutions

Assess security risks in third-party components. Determine how effectively a technology is likely to alleviate risks. Identify lingering security risks in chosen security technologies.

Annotate class designs with security properties

Elaborate security policies for individual data fields.

Specify database security configuration (Database Designer)

Define a secure default configuration for database resources that are deployed as part of an implementation. Identify a recommended configuration for database resources for databases that are deployed by a third party.

Address reported security issues

Ensure that identified security risks in an implementation are properly considered.

ISSES

Co-funded by the Erasmus+ Programme of the European Union

Project Manager	Requirements Specifier	Architect	Designer	Implementer	Test Analyst	Security Auditor

Implement interface contracts

Provide unit-level semantic input validation. Identify reliability errors in a structured way at the earliest opportunity.

Implement and elaborate resource policies and security technologies

Implement security functionality to specification.

Build operational security guide (Integrator)

Provide stakeholder with documentation on operational security measures that can better secure the product. Provide documentation for the use of security functionality within the product.

Project Manager	Requirements Specifier	Architect	Designer	Implementer	Test Analyst	Security Auditor

Identify, implement and perform security tests

Find security problems not detected by implementation review. Find security risks introduced by the operational environment. Act as a defense-in-depth mechanism, catching failures in design, specification, or implementation.

Verify security attributes of resources (Tester)

Confirm that software conforms to previously defined security policies.

Project Manager	Requirements Specifier	Architect	Designer	Implementer	Test Analyst	Security Auditor

Perform security analysis of system requirements and design (threat modelling)

Assess likely system risks timely and cost-effectively by analysing the requirements and design. Identify high-level system threats that are not documented in requirements or supplemental documentation. Identify inadequate or improper security requirements. Assess the security impact of non-security requirements.

Perform source-level security review

Find security vulnerabilities introduced into implementation.

Студија случаја: NIST SSDF

Радни оквир за развој безбедног софтвера

ISSES

Co-funded by the Erasmus+ Programme of the European Union

Радни оквир сигурног развоја

Објављен од стране NIST-а пре 2 године

Радни оквир који помаже да се произведе безбеднији софтвер

Заснован на установљеним документима са добрим праксама за развој безбедног софтвера

Размотрићемо препоручене праксе, посебно:

- **Шта која пракса подразумева**
- **Како се свака пракса мапира на конкретне задатке**

ISSES Co-funded by the Erasmus+ Programme of the European Union

Prepare the Organization	Protect the Software	Produce Well-Secured Software	Respond to Vulnerabilities

Define Security Requirements for Software Development (PO.1)

- PO.1.1: Identify all applicable security requirements for the organization’s general software development, and maintain the requirements over time.

Implement Roles and Responsibilities (PO.2)

- PO.2.1: Create new roles and alter responsibilities for existing roles to encompass all parts of the SSDF. Periodically review the defined roles and responsibilities, and update them as needed.
- PO.2.2: Provide role-specific training for all personnel with responsibilities that contribute to secure development. Periodically review role-specific training and update it as needed.
- PO.2.3: Obtain upper management commitment to secure development, and convey that commitment to all with SSDF-related roles and responsibilities.

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Prepare the Organization	Protect the Software	Produce Well-Secured Software	Respond to Vulnerabilities

Implement a Supporting Toolchain (PO.3)

- PO.3.1: Specify which tools or tool types are to be included in each toolchain and which are mandatory, as well as how the toolchain components are to be integrated with each other.
- PO.3.2: Following sound security practices, deploy and configure tools, integrate them within the toolchain, and maintain the individual tools and the toolchain as a whole.
- PO.3.3: Configure tools to collect evidence and artifacts of their support of the secure software development practices

Define Criteria for Software Security Checks (PO.4)

- PO.4.1: Define criteria for software security checks throughout the SDLC.
- PO.4.2: Implement processes, mechanisms, etc. to gather the necessary information in support of the criteria.

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Prepare the Organization	Protect the Software	Produce Well-Secured Software	Respond to Vulnerabilities

Protect All Forms of Code from Unauthorized Access and Tampering (PS.1)

- PS.1.1: Store all forms of code, including source code and executable code, based on the principle of least privilege so that only authorized personnel have the necessary forms of access.

Provide a Mechanism for Verifying Software Release Integrity (PS.2)

- PS.2.1: Make verification information available to software consumers.

Archive and Protect Each Software Release (PS.3)

- PS.3.1: Securely archive a copy of each release and all of its components (e.g., code, package files, third-party libraries, documentation), and release integrity verification information.

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Prepare the Organization	Protect the Software	Produce Well-Secured Software	Respond to Vulnerabilities

Design Software to Meet Security Requirements and Mitigate Security Risks (PW.1)

- PW.1.1: Use forms of risk modelling, such as threat modelling, attack modelling, or attack surface mapping, to help assess the security risk for the software.

Review the Software Design to Verify Compliance with Security Requirements and Risk Information (PW.2)

- PW.2.1: Have a qualified person who was not involved with the software design review it to confirm that it meets all of the security requirements and satisfactorily addresses the identified risk information.

Verify Third-Party Software Complies with Security Requirements (PW.3)

- PW.3.1: Communicate requirements to third parties who may provide software modules and services to the organization for reuse by the organization's own software.

- PW.3.2: Use appropriate means to verify that commercial, open source, and all other third-party software modules and services comply with the requirements.

Reuse Existing, Well-Secured Software When Feasible Instead of Duplicating Functionality (PW.4)

- PW.4.1: Acquire well-secured components (e.g., software libraries, modules, middleware, frameworks) from third parties for use by the organization's software.
- PW.4.2: Create well-secured software components in-house following SDLC processes to meet common internal software development needs that cannot be better met by third-party software.
- PW.4.3: Where appropriate, build in support for using standardized security features and services (e.g., integrating with log management, identity management, access control, and vulnerability management systems) instead of creating proprietary implementations of security features and services.

Create Source Code Adhering to Secure Coding Practices (PW.5)

- PW.5.1: Follow all secure coding practices that are appropriate to the development languages and environment.
- PW.5.2: Have the developer review their own human-readable code, analyze their own human-readable code, and/or test their own executable code to complement (not replace) code review, analysis, and/or testing performed by others.

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Prepare the Organization	Protect the Software	Produce Well-Secured Software	Respond to Vulnerabilities

Configure the Compilation and Build Processes to Improve Executable Security (PW.6)

- PW.6.1: Use compiler and build tools that offer features to improve executable security.
- PW.6.2: Determine which compiler and build tool features should be used and how each should be configured, then implement the approved configuration for compilation and build tools, processes, etc.

Review and/or Analyze Human-Readable Code to Identify Vulnerabilities and Verify Compliance with Security Requirements (PW.7)

- PW.7.1: Determine whether code review (i.e., a person directly looks at the code to find issues) and/or code analysis (i.e., tools are used to find issues in code, either in a fully automated way or in conjunction with a person) should be used.
- PW.7.2: Perform the code review and/or code analysis based on the organization’s secure coding standards, and document and triage all discovered issues and recommended remediations in the development team’s workflow or issue tracking system.

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Prepare the Organization	Protect the Software	Produce Well-Secured Software	Respond to Vulnerabilities

Test Executable Code to Identify Vulnerabilities and Verify Compliance with Security Requirements (PW.8)

- PW.8.1: Determine if executable code testing should be performed and, if so, which types should be used.
- PW.8.2: Design the tests, perform the testing, and document the results.

Configure the Software to Have Secure Settings by Default (PW.9)

- PW.9.1: Determine how to configure each setting that has an effect on security so that the default settings are secure and do not weaken the security functions provided by the platform, network infrastructure, or services.
- PW.9.2: Implement the default settings (or groups of default settings, if applicable), and document each setting for software administrators.

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Prepare the Organization	Protect the Software	Produce Well-Secured Software	Respond to Vulnerabilities

Identify and Confirm Vulnerabilities on an Ongoing Basis (RV.1)

- RV.1.1: Gather information from consumers and public sources on potential vulnerabilities in the software and any third-party components that the software uses, and investigate all credible reports.
- RV.1.2: Review, analyze, and/or test the software's code to identify or confirm the presence of previously undetected vulnerabilities.
- RV.1.3: Have a team and process in place to handle the responses to vulnerability reports and incidents.

Assess, Prioritize, and Remediate Vulnerabilities (RV.2)

- RV.2.1: Analyze each vulnerability to gather sufficient information to plan its remediation.
- RV.2.2: Develop and implement a remediation plan for each vulnerability.

ISSES

Co-funded by the Erasmus+ Programme of the European Union

Prepare the Organization | Protect the Software | Produce Well-Secured Software | Respond to Vulnerabilities

Analyze Vulnerabilities to Identify Their Root Causes (RV.3)

- RV.3.1: Analyze all identified vulnerabilities to determine the root cause of each vulnerability.
- RV.3.2: Analyze the root causes over time to identify patterns, such as when a particular secure coding practice is not being followed consistently.
- RV.3.3: Review the software for other instances of the reported problem and proactively fix them rather than waiting for external reports.
- RV.3.4: Review the SDLC process, and update it as appropriate to prevent (or reduce the likelihood of) the root cause recurring in updates to this software or in new software that is created.

ISSES

Co-funded by the Erasmus+ Programme of the European Union

Студија случаја: OWASP SAMM

Модел зрелости безбедног софтвера

ISSES

Co-funded by the Erasmus+ Programme of the European Union

Модел зрелости сигурног развоја

Објављен и ажуриран од стране OWASP-а (тренутно на верзији 2)

Модел зрелости који помаже да се произведе безбеднији софтвер

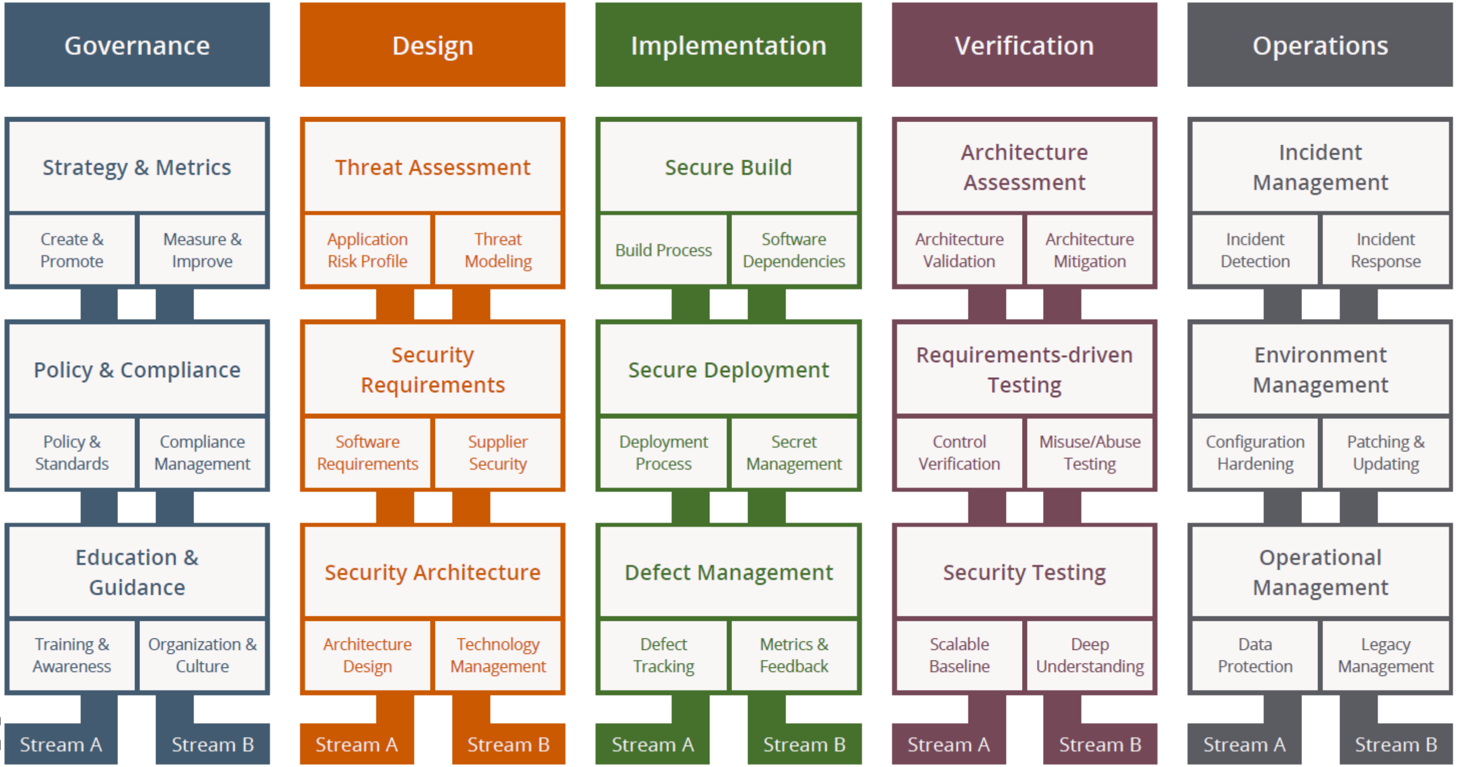
Дефинише мерљив начин провере безбедности софтвера у било којој организацији

Размотрићемо модел, посебно:

- **Шта која пословна активност подразумева**
- **Како се свака активност мапира на ниво зрелости**

ISSES Co-funded by the Erasmus+ Programme of the European Union

SAMM модел



of the European Union

Design – Security Requirements – Software Requirements

Software requirements specify objectives and expectations to protect the service and data at the core of the application.

- **Level 1:** High-level application security objectives are mapped to functional requirements.
- **Level 2:** Structured security requirements are available and utilized by developer teams.
- **Level 3:** Build a requirements framework for product teams to utilize.

ISSES Co-funded by the Erasmus+ Programme of the European Union

Software Requirements Measurement – Level 1

Do project teams specify security requirements during development?

Teams derive security requirements from functional requirements and customer or organization concerns

Security requirements are specific, measurable, and reasonable

Security requirements are in line with the organizational baseline

ISSES Co-funded by the Erasmus+ Programme of the European Union

Software Requirements Measurement – Level 2

Do you define, structure, and include prioritization in the artifacts of the security requirements gathering process?

Security requirements take into consideration domain specific knowledge when applying policies and guidance to product development

Domain experts are involved in the requirements definition process

You have an agreed upon structured notation for security requirements

Development teams have a security champion dedicated to reviewing security requirements and outcomes

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Software Requirements Measurement – Level 3

Do you use a standard requirements framework to streamline the elicitation of security requirements?

A security requirements framework is available for project teams

The framework is categorized by common requirements and standards-based requirements

The framework gives clear guidance on the quality of requirements and how to describe them

The framework is adaptable to specific business requirements

ISSES Co-funded by the Erasmus+ Programme of the European Union

ОБАВЕШТЕЊЕ ЗА СТУДЕНТЕ

Настава на предмету Развој безбедног софтвера подразумева изучавање различитих механизама којима се нарушава информациона безбедност и врше напади на интернет апликације и софтверске системе.

Примена ових механизама када се извршавају према системима физичких и правних лица која нису упозната и сагласна са активностима на провери рањивости и тестирању упада у њихове системе је кажњива према Кривичном законику Републике Србије (Чланови 298 до 304а).

Студенти на предмету Развој безбедног софтвера могу ове методе за потребе изучавања да користе искључиво у оквиру затвореног лабораторијског окружења које је обезбеђено за наставу на предмету Развој безбедног софтвера.

Студенти не могу да подразумевају да су на било који начин охрабрени од стране наставника или да им се препоручује да користе ове методе који се изучавају према другим апликацијама Електротехничког факултета или апликацијама било ког трећег правног или физичког лица.

Свака евентуална активност коју би предузео неки студент коришћењем ових метода и механизама према апликацијама које нису у оквиру лабораторије на предмету искључива је одговорност студента.

ISSES

Co-funded by the Erasmus+ Programme of the European Union