Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Computer Systems Software Department

SOFTWARE SECURITY

Working program of the academic discipline (Syllabus)

Requisites of the Course

Cycle of Higher Education	***First (Bachelor)***
Field of Study	*12 Information Technologies*
Specialty	*121 Software Engineering*
Education Program	*Software Engineering of Multimedia and Information Retrieval Systems*
Type of Course	*Normative*
Mode of Study	*Full-time*
Year of Study, Semester	*4th year, 7th semester*
ECTS workload	*Lectures: 36 hours., laboratory work: 18 hours, self-study: 66 hours*
Testing and Assessment	*Exam, modular control work, calendar control*
Class Schedule	*According to the schedule for the autumn semester of the current academic year (schedule.kpi.ua)*
Language of Instructions	*English*
Course Instructors	Lecturer: *PhD, assistant, Severin Andrii,* severinandrey97@gmail.com Laboratory work: *PhD, assistant, Severin Andrii, severinandrey97@gmail.com*
Access to the Course	Google classroom: https://classroom.google.com/

Outline of the Course

1. Course description, goals, objectives, and learning outcomes

The study of the "Software Security" discipline allows students to develop the competencies necessary for solving practical tasks of professional activity related to the development of software in terms of its security (primarily confidentiality, integrity, availability).

*The **purpose** of studying the "Software Security" discipline is the formation of students' abilities to independently develop software by defining and implementing its security requirements (primarily confidentiality, integrity, availability).*

*The **subject** of the "Software Security" discipline is methods of modeling software security threats.*

*The study of the "Software Security" discipline forms **general competencies (GC)** in students, necessary for solving practical tasks of professional activities related to the development, improvement and operation of software:*

***GC08** Ability to act on the basis of ethical considerations.*

***GC10** Ability to act socially responsibly and consciously.*

*The study of the "Software Security" discipline forms **professional competencies (PC)**, in students, necessary for solving practical tasks of professional activities related to the development, improvement and operation of software:*

***PC01** Ability to identify, classify and formulate software requirements.*

***PC03** Ability to develop architectures, modules and components of software systems.*

PC06 *Ability to analyze, select and apply methods and tools to ensure information security (including cybersecurity).*

PC08 *Ability to apply fundamental and interdisciplinary knowledge to successfully solve software engineering problems.*

PC14 *Aptitude for algorithmic and logical thinking.*

The study of the "Software Security" discipline contributes to the formation in students of the following ***program learning outcomes (PLO)*** *according to the educational program:*

PLO01 *To analyze, purposefully search for and select the information and reference resources and knowledge necessary for solving professional tasks, taking into account modern achievements of science and technology.*

PLO02 *To know the code of professional ethics, understand the social significance and cultural aspects of software engineering and adhere to them in professional activities.*

PLO05 *To know and apply relevant mathematical concepts, methods of domain, system and object-oriented analysis and mathematical modeling for software development.*

PLO13 *To know and apply methods of developing algorithms, designing software, data and knowledge structures.*

PLO18 *To know and be able to apply information technologies for data processing, storage and transmission.*

PLO21 *To know, analyze, choose, competently apply the means of ensuring information security (including cybersecurity) and data integrity in accordance with the applied tasks being solved and the software systems being created.*

PLO30 *To be able to apply programming technologies for the development of software for multimedia and information retrieval systems.*

2. Pre-requisites and post-requisites of the discipline (place in the structural and logical scheme of training according to the relevant educational program)

The successful study of the "Software Security" discipline is preceded by the study of the "Databases", "Programming", "Components of Software Engineering" disciplines of the curriculum of bachelor's training in specialty 121 Software Engineering.

The theoretical knowledge and practical skills obtained during the mastering of the "Software Security" discipline ensure the successful implementation of course projects and diploma projects in the specialty 121 Software Engineering.

3. Content of the course

The "Software Security" discipline involves the study of the following topics:

Topic 1. Introduction to software security

Topic 2. Methods of modeling software security threats

Topic 3. Basic principles of developing secure software

Modular control work

Exam

4. Educational materials and resources

Basic literature:

1. Security of the software environment. Electronic campus of NTUU "KPI named after Igor Sikorsky".

Materials from the discipline "Security of the software environment". – Access to registered students.

Additional literature:

2. Shostack A. Threat Modeling: Designing for Security. Indianapolis: John Wiley & Sons, 2014. 590 p.

3. Tarandach I., Coles M. J. Threat Modeling. A Practical Guide for Development Teams. Sebastopol: O'Reilly Media, 2020, 201 p.

4. Threat Modeling. URL: https://owasp.org/www-community/Threat_Modeling (accessed on: 01.04.2025).

5. Common Vulnerability Scoring System v3.1: Specification Document. URL: https://www.first.org/cvss/v3.1/specification-document (accessed on: 01.04.2025).

6. LINDDUN framework. URL: https://www.linddun.org/linddun (accessed on: 01.04.2025).

7. ISO/IEC 27005:2022. Information security, cybersecurity and privacy protection - Guidance on managing information security risks. URL: https://www.iso.org/standard/80585.html (accessed on: 01.04.2025).

8. MITER ATT&CK. URL: https://attack.mitre.org/ (accessed on: 01.04.2025).

9. Threat Modeling Manifesto. URL: https://www.threatmodelingmanifesto.org/ (accessed on: 01.04.2025).

10. Threat Modeling. URL: https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling (accessed on: 01.04.2025).

11. Threat Modeling. Process. URL: https://owasp.org/www-community/Threat_Modeling_Process (accessed on: 01.04.2025).

12. Create a threat model using data-flow diagram elements. URL: https://docs.microsoft.com/en-us/learn/modules/tm-create-a-threat-model-using-foundational-data-flow-diagram-elements/ (accessed on: 01.04.2025).

13. ISO/IEC 27000:2018. Information technology. Security techniques. Information security management systems. Overview and vocabulary. [Valid from 2018-02-07]. URL: https://www.iso.org/standard/73906.html (accessed on: 01.04.2025).

14. DREAD Threat Modeling: An Introduction to Qualitative Risk Analysis. URL: https://www.eccouncil.org/cybersecurity-exchange/threat-intelligence/dread-threat-modeling-intro/ (accessed on: 01.04.2025).

15. Schneier B. Attack Trees. URL: https://www.schneier.com/academic/archives/1999/12/attack_trees.html (accessed on: 01.04.2025).

16. An Alternative: Attack Trees. URL: https://www.oreilly.com/library/view/building-secure-servers/0596002173/ch01s03.html (accessed on: 01.04.2025)

17. Common Vulnerability Scoring System v3.1: Examples. URL: https://www.first.org/cvss/v3.1/examples (accessed on: 01.04.2025).

18. IEC 31010:2019. Risk management. Risk assessment techniques. [Valid from 2019-06-17]. URL: https://www.iso.org/standard/72140.html (accessed on: 01.04.2025).

19. Finding Cyber Threats with ATT&CK™-Based Analytics. URL: https://www.mitre.org/sites/default/files/2021-11/16-3713-finding-cyber-threats-with-attack-based-analytics.pdf (accessed on: 01.04.2025).

20. OWASP Top 10:2021. URL: https://owasp.org/Top10/ (accessed on: 01.04.2025).

21. OWASP Secure Coding Practices Quick Reference Guide. URL: https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/assets/docs/OWASP_SCP_Quick_Reference_Guide_v21.pdf (accessed on: 01.04.2025).

Use to master the practical skills of the discipline. The materials are freely available on the Internet.

Educational content

5. Methods of mastering an educational discipline (educational component

No.	Type of training session	Description of the training session
	Topic 1. Introduction to software security	
1	*Lecture 1. Course content, introduction to software security*	*Overview of course content. The concept of software security. Confidentiality, integrity, and availability properties. Secure software development life cycle. Approaches to defining software security requirements. Open Web Application Security Project (OWASP)* *Tasks on self-study: Item 6 No. 1.*
2	*Lecture 2. Modeling software security threats*	*The concept of software security threat modeling. The process of modeling software security threats. Stages of modeling software security threats. Software decomposition in terms of threats.* *Tasks on self-study: Item 6 No. 2.*
3	*Laboratory work 1. Software decomposition in terms of threats*	*Task: perform a software decomposition in terms of threats.* *Tasks on self-study: Item 6 No. 3.*
	Topic 2. Methods of modeling software security threats	
4	*Lecture 3. A method of modeling software security threats based on a data flow diagram*	*The concept and characteristics of the data flow diagram. Features of using a data flow diagram to model software security threats. Elements of a data flow diagram. The process of building a data flow diagram. Rules for constructing a data flow diagram.* *Tasks on self-study: Item 6 No. 4.*
5	*Lecture 4. Stages of building a data flow diagram*	*Stages of building a data flow diagram. Definition of software processes. Definition of software data stores. Definition of software entities. Defining data flows and trust boundaries between software elements.* *Tasks on self-study: Item 6 No. 5.*
6	*Laboratory work 2. Creating a model of software security threats based on a data flow diagram*	*Task: create a software security threat model based on a data flow diagram.* *Tasks on self-study: Item 6 No. 6.*
7	*Lecture 5. STRIDE software security threat modeling method*	*Characteristics of the STRIDE method. Attributes of the STRIDE method: spoofing, falsification, failure, disclosure, denial of service, privilege escalation. The process of modeling software security threats using the STRIDE*

		method. *Tasks on self-study: Item 6 No. 7.*
8	*Lecture 6. Stages of modeling software security threats using the STRIDE method*	*Defining the threat of spoofing. Definition of the threat of falsification. Determining the threat of failure. Determination of the threat of information disclosure. Determining the threat of denial of service. Identifying the threat of privilege escalation. Defining software security requirements.* *Tasks on self-study: Item 6 No. 8.*
9	*Laboratory work 3. Creating a model of software security threats using the STRIDE method*	*Task: create a model of software security threats using the STRIDE method.* *Tasks on self-study: Item 6 No. 9.*
10	*Lecture 7. DREAD software security threat modeling method*	*Characteristics of the DREAD method. DREAD threat assessment scales. Selection of the threat assessment scale by the DREAD method. The process of modeling software security threats using the DREAD method.* *Tasks on self-study: Item 6 No. 10.*
11	*Lecture 8. Stages of modeling software security threats using the DREAD method*	*Identifying software security threats. Definition of software security threat assessment scale. Software security threat assessment. Software security threat ranking. Defining software security requirements.* *Tasks on self-study: Item 6 No. 11.*
12	*Laboratory work 4. Creating a model of software security threats using the DREAD method*	*Task: create a model of software security threats using the DREAD method.* *Tasks on self-study: Item 6 No. 12.*
13	*Lecture 9. Modeling software security threats by creating an attack tree*	*Characteristics of the attack tree creation method. Ways to use the attack tree. Choosing how to use the attack tree. The process of modeling software security threats using the method of creating an attack tree.* *Tasks on self-study: Item 6 No. 13.*
14	*Lecture 10. Stages of modeling software security threats using the method of creating an attack tree*	*Choosing how to display the attack tree. Creating the root node of the attack tree. Creating subnodes of the root node of the attack tree. Checking the completeness of the created attack tree. Pruning the generated attack tree. Checking the generated attack tree.* *Tasks on self-study: Item 6 No. 14.*
15	*Laboratory work 5. Creating a model of software security threats using the attack tree method*	*Task: create a model of software security threats using the attack tree method.* *Tasks on self-study: Item 6 No. 15.*
16	*Lecture 11. The method of assessing the severity of software vulnerabilities*	*Characteristics of the method of evaluating the severity of vulnerabilities. Metrics for evaluating the severity of vulnerabilities. A string vector for assessing the severity of vulnerabilities. Vulnerability severity calculator.*

		Tasks on self-study: Item 6 No. 16.
17	*Lecture 12. Stages of the method of assessing the severity of software vulnerabilities*	*Definition of basic metrics. Determination of time metrics. Defining user environment metrics. Definition of the equation for evaluating the severity of software vulnerabilities. Using the software vulnerability severity calculator.* *Tasks on self-study: Item 6 No. 17.*
18	*Laboratory work 6. Evaluation of software vulnerabilities according to the CVSS standard*	*Task: evaluate software vulnerabilities according to the CVSS standard.* *Tasks on self-study: Item 6 No. 18.*
19	*Lecture 13. Modeling software security threats using the LINDDUN method. Detection of software privacy threats*	*Characteristics of the LINDDUN method. Categories of threats according to the LINDDUN method. Building a software security threat model. Detection of software privacy threats. Managing software privacy threats. Mapping the elements of the data flow diagram in relation to categories of privacy threats. Identifying and documenting privacy threats.* *Tasks on self-study: Item 6 No. 19.*
20	*Lecture 14. Methods of software security risk assessment. Assessment of information security risks using the "Consequences - Probability Matrix" method*	*Varieties of software security risk assessment methods. Criteria for choosing software security risk assessment methods. Approaches to the selection of software security risk assessment methods. The "Consequences - Probability Matrix" method. Characteristics of the method, a risk assessment scale, risk acceptability criteria, stages of using the method.* *Tasks on self-study: Item 6 No. 20.*
21	*Modular control work. Implementation of software security requirements*	*Task: implement software security requirements.* *Tasks on self-study: Item 6 No. 21.*
22	*Lecture 15. MITER ATT&CK knowledge base on software hacker tactics and techniques*	*Structure of the MITER ATT&CK knowledge base. MITER ATT&CK matrix. Categories of structuring knowledge about tactics and techniques of the violator: enterprise, mobile devices, industrial control systems. Determination of the offender's behavior. Determination of probable scenarios of the violator's actions. Assessment of likely actions of the violator.* *Tasks on self-study: Item 6 No. 22.*
Topic 3. Basic principles of developing secure software		
23	*Lecture 16. Software security requirements. Principles of formulating qualitative security requirements.*	*The place of software security requirements in the software development process. Principles of formulating qualitative security requirements: unambiguousness, verifiability, clarity, correctness, understandability, feasibility, independence, atomicity, necessity.* *Tasks on self-study: Item 6 No. 23.*

24	*Laboratory work 7. Implementing software security requirements (part 1)*	*Task: demonstrate implemented software security requirements (part 1).* *Tasks on self-study: Item 6 No. 24.*
25	*Lecture 17. Common software vulnerabilities and methods of protection against them*	*A review of common software vulnerabilities and methods of protection against them. Injections, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), Man-in-the-Middle (MitM).* *Tasks on self-study: Item 6 No. 25.*
26	*Lecture 18. Best practices for developing secure software*	*A review of best practices for avoiding common software vulnerabilities. Principles of secure coding: privilege minimization, security testing, secure configuration, software updates, logging, and monitoring.* *Tasks on self-study: Item 6 No. 26.*
27	*Laboratory work 7. Implementing software security requirements (part 2)*	*Task: demonstrate implemented software security requirements (part 2).* *Tasks on self-study: Item 6 No. 27.*

6. Self-study

The "Software Security" discipline is based on self-study for classes on theoretical and practical topics.

No.	*Topic for self-studying*	*Number of hours*	*Literature*
1	*Preparation for lecture 1*	*1*	*1; 2; 3; 4; 9–11; 21*
2	*Preparation for lecture 2*	*1*	*1; 2; 3; 4; 9–11*
3	*Preparation for laboratory work 1*	*2*	*1; 2; 3; 4; 9–11*
4	*Preparation for lecture 3*	*1*	*1; 2; 3; 4; 9–11*
5	*Preparation for lecture 4*	*1*	*1; 2; 3; 4; 9–11*
6	*Preparation for laboratory work 2*	*2*	*1; 2; 3; 4; 9–11*
7	*Preparation for lecture 5*	*1*	*1; 2; 3; 4; 9–11; 13*
8	*Preparation for lecture 6*	*1*	*1; 2; 3; 4; 9–11; 13*
9	*Preparation for laboratory work 3*	*2*	*1; 2; 3; 4; 9–11; 13*
10	*Preparation for lecture 7*	*1*	*1; 2; 3; 4; 9–11; 14*
11	*Preparation for lecture 8*	*1*	*1; 2; 3; 4; 9–11; 14*
12	*Preparation for laboratory work 4*	*2*	*1; 2; 3; 4; 9–11; 14*
13	*Preparation for lecture 9*	*1*	*1; 2; 3; 4; 9–11; 15; 16*
14	*Preparation for lecture 10*	*1*	*1; 2; 3; 4; 9–11; 15; 16*
15	*Preparation for laboratory work 5*	*2*	*1; 2; 3; 4; 9–11; 15; 16*
16	*Preparation for lecture 11*	*1*	*1; 2; 3; 4; 5; 9–11; 17*

17	*Preparation for lecture 12*	*1*	*1; 2; 3; 4; 5; 9–11; 17*
18	*Preparation for laboratory work 6*	*2*	*1; 2; 3; 4; 5; 9–11; 17*
19	*Preparation for lecture 13*	*1*	*1; 2; 3; 4; 6; 9–11*
20	*Preparation for lecture 14*	*1*	*1; 2; 3; 4; 7; 9–11; 18*
21	*Preparation for modular control work*	*4*	*1; 2; 3; 4; 9–16*
22	*Preparation for lecture 15*	*1*	*1; 2; 3; 4; 8–11; 19*
23	*Preparation for lecture 16*	*1*	*1; 2; 3; 4; 7; 9–11; 20*
24	*Preparation for laboratory work 7 (part 1)*	*1*	*1; 2; 3; 4; 9–16*
25	*Preparation for lecture 17*	*1*	*1; 2; 3; 4; 20*
26	*Preparation for lecture 18*	*1*	*1; 2; 3; 4; 21*
27	*Preparation for laboratory work 7 (part 2)*	*1*	*1; 2; 3; 4; 9–16*
28	*Preparation for the exam*	*30*	*1-21*

Policy and Control

7. Policy of academic discipline (educational component)

Attending lectures is mandatory.

Attending laboratory work classes may be occasional and as needed for consultation/protection of laboratory work.

Rules of behavior in classes: activity, respect for those present, turning off phones.

Adherence to the policy of academic integrity.

Rules for the protection of laboratory work: the work must be performed in accordance with the assigned tasks and according to the option chosen by the student.

8. Types of control and rating system for evaluating learning outcomes

*During the semester, students perform **7 laboratory works**. The maximum number of points for each laboratory work: 5 points.*

Points are awarded for:
- the quality of performance and protection of laboratory work: 0-4 points.
- timely presentation of work for defense: 0-1 point.

Criteria for evaluating the quality of performance and protection:
4 points – the work is done qualitatively, in full, the answers are complete, well-argued;
3 points – the work is done qualitatively, in full, but has shortcomings, answers with minor errors;
1-2 points – the work is done with sufficient quality, in full, but contains significant shortcomings, answers with significant errors;
0 points – the work is done poorly, not in full, the answers are either absent or incorrect.

***The maximum number of points for performing and defending laboratory work**:*
5 points × 7 laboratory works = 35 points.

*The task of **modular control work** is to implement software security requirements.*
The answer is evaluated by 15 points.

Criteria for evaluating a modular control work:

13–15 points – the answer is correct, complete, well-argued;
8–12 points – in general, the answer is correct, but it has flaws;
1-7 points – there are significant errors in the answer;
0 points – there is no answer or the answer is incorrect.

***The maximum number of points for a modular control work**:*
15 points × 1 task = 15 points.

***The rating scale for the discipline is equal to**:*
$R = R_S = R_{lab.\ works} + R_{modular\ control\ work} + R_{exam} = 35$ points + 15 points + 50 points = 100 points

Calendar control: conducted twice a semester as a monitoring of the current state of fulfillment of the syllabus requirements.
At the first certification (7th week), the student receives "Passed" if his current rating is at least 10 points (50% of the maximum number of points that the student can receive before the first certification).
At the second certification (13th week), the student receives "Passed" if his current rating is at least 20 points (50% of the maximum number of points that the student can receive before the second certification).

Semester control: exam.
Conditions for admission to semester control:
A prerequisite for a student's admission to the exam is a semester rating (R_S) of at least 30 points.
After passing the exam, a grade is assigned according to the table (Table of correspondence of rating points to grades on the university scale).

The exam task consists of 3 questions - 2 theoretical and 1 practical. The answer to each theoretical question is worth 15 points, and the answer to a practical question is worth 20 points.
Evaluation criteria for a theoretical question:
13–15 points – the answer is correct, complete, well-argued;
8–12 points – in general, the answer is correct, but it has flaws;
1–7 points – there are significant errors in the answer;
0 points – there is no answer or the answer is incorrect.

Evaluation criteria for a practical question:
16–20 points – the answer is correct, complete, well-argued;
9–15 points – in general, the answer is correct, but it has flaws;
1–8 points – there are significant errors in the answer;
0 points – there is no answer or the answer is incorrect.

Table of correspondence of rating points to grades on the university scale:

Score	*Grade*
100-95	Excellent
94-85	Very good
84-75	Good
74-65	Satisfactorily
64-60	Enough
Less than 60	Unsatisfactorily
Admission conditions not met	Not allowed

9. Additional information on the discipline (educational component)

The list of questions submitted for semester control is given in Appendix 1.

Working program of the academic discipline (syllabus):

Is designed by PhD, assistant, Severin Andrii

Adopted by Computer Systems Software Department (protocol № 15 from 15.04.2026)

Approved by the Faculty Board of Methodology (protocol № 11 from 22.05.2026)

Appendix 1. List of questions submitted for semester control

1. Concept of software security.
2. Confidentiality, integrity, and availability properties of software data.
3. Secure software development life cycle.
4. Approaches to defining software security requirements.
5. Concept of software security threat modeling.
6. Process of modeling software security threats.
7. Stages of software security threat modeling.
8. Software decomposition in terms of threats.
9. Method of modeling software security threats based on a data flow diagram.
10. STRIDE software security threat modeling method.
11. DREAD software security threat modeling method.
12. Modeling of software security threats by the method of creating an attack tree.
13. Method of assessing the severity of software vulnerabilities.
14. Severity metrics of software vulnerabilities.
15. Simulation of software security threats using the LINDDUN method.
16. Approaches to the selection of software security risk assessment methods.
17. Assessment of information security risks using the "Consequences - Probability Matrix" method.
18. Creation of a software security threat model based on the MITER ATT&CK knowledge base.
19. Determination of probable scenarios of actions of a software security violator based on the MITER ATT&CK knowledge base.
20. Software security requirements. Principles of formulating qualitative security requirements.
21. Common software vulnerabilities and methods of protection against them.
22. Practices for developing secure software. Principles of secure coding.