SYLLABUS

Advanced Software Security

University year 2025-2026

1. Information regarding the programme

1.1. Higher education institution	Babeș-Bolyai University
1.2. Faculty	Mathematics and Computer Science
1.3. Department	Computer Science
1.4. Field of study	Computer Science
1.5. Study cycle	Master
1.6. Study programme/Qualification	Cyber Security
1.7. Form of education	Full time

2. Information regarding the discipline

2.1. Name of the discipline	**Advanced Software Security**					Discipline code	**MME8199**
2.2. Course coordinator					Assoc.prof.phd. Mihai SUCIU		
2.3. Seminar coordinator					Assoc.prof.phd. Mihai SUCIU		
2.4. Year of study	2	2.5. Semester	3	2.6. Type of evaluation	E	2.7. Discipline regime	Mandatory

3. Total estimated time (hours/semester of didactic activities)

3.1. Hours per week	**4**	of which: 3.2 course	**2**	3.3 seminar/laboratory/project	**2**
3.4. Total hours in the curriculum	56	of which: 3.5 course	28	3.6 seminar/laboratory/project	**28**

Time allotment for individual study (ID) and self-study activities (SA)	**hours**
Learning using manual, course support, bibliography, course notes (SA)	30
Additional documentation (in libraries, on electronic platforms, field documentation)	20
Preparation for seminars/labs, homework, papers, portfolios and essays	35
Tutorship	4
Evaluations	5
Other activities:	

3.7. Total individual study hours	**94**
3.8. Total hours per semester	**150**
3.9. Number of ECTS credits	**6**

4. Prerequisites (if necessary)

4.1. curriculum	• Computer System Architecture • Operating Systems • Data Structures and Algorithms • Data Bases • Web Programming
4.2. competencies	• Programming in C, basic knowledge of Intel x86 architecture, basic knowledge of web programming and SQL

5. Conditions (if necessary)

5.1. for the course	• course room with video projector
5.2. for the seminar /lab activities	• laboratory room with computers

6.1. Specific competencies acquired [1]

Professional/essential competencies	• Know and understand the main paradigms related to data protection: confidentiality, integrity and data availability. • Learning how the main forms of malware and the main forms of attacks on the Internet work, as well as the methods of protection against them..
Transversal competencies	• Professional communication skills; concise and precise description, both oral and written, of professional results. • Ethic and fair behavior, commitment to professional deontology.

6.2. Learning outcomes

Knowledge	The student knows which are the best security mechanisms that can be implemented on the Internet; acquires knowledge about the worst types of vulnerabilities in the field, as well as about measures to prevent these vulnerabilities.
Skills	The student is able to identify possible security issues in software systems, has the ability to evaluate the security features of software applications at the source code level.
Responsibility and autonomy:	The student has the ability to work independently to identify vulnerabilities and security threats to an organization or business.

7. Objectives of the discipline (outcome of the acquired competencies)

7.1 General objective of the discipline	• Ability to evaluate the security features of a software application based on the source code. Acquiring the minimum basic skills of writing a source code without vulnerability.

[1] One can choose either competences or learning outcomes, or both. If only one option is chosen, the row related to the other option will be deleted, and the kept one will be numbered 6.

7.2 Specific objective of the discipline	• Knowledge of the basic mechanisms that define the security of the system and the software environment in which an application runs (i.e. the security model), such as: access permissions, security policies, interaction with the external environment, etc. • Knowledge of the main types of software vulnerabilities, such as: use of incorrectly validated user data, uncontrolled direct or indirect interaction with the external environment of the application, etc. • Learning effective techniques for studying and evaluating source code from a security perspective and the ability to identify possible vulnerabilities. • Ability to assess the implications of a discovered vulnerability. • Knowledge of techniques and function libraries useful in writing a source code without vulnerabilities and the ability to use them in real situations.

8. Content

8.1 Course	Teaching methods	Remarks
Concepts and basics related to software vulnerabilities and methods and tools for developing software without vulnerabilities and evaluating software from the perspective of possible vulnerabilities	Exposure: description, explanation, examples, debate	
Memory corruption vulnerabilities (buffer / integer overflow, etc.)		
Vulnerabilities specific to the C language: arithmetic limits (representation), type conversions, pointers, etc.		
Vulnerabilities in the structural components of a software application (Program building blocks)		
Vulnerabilities in the use and manipulation of strings and metacharacters		
Vulnerabilities specific to UNIX operating systems		
Synchronization vulnerabilities		
Web vulnerabilities: SQL code injection, XSS, XSRF etc.		
Cryptography vulnerabilities: vulnerable passwords, predictable random numbers, etc.		
Proactive approaches to security		
Proactive approaches to security		
Proactive approaches to security		
Proactive approaches to security		
Proactive approaches to security		
Bibliography		
8.2 Seminar / laboratory	Teaching methods	Remarks
Tools useful in identifying and assessing vulnerabilities in a source code: source code browsers, debuggers, executable code browsers (binary), fuzzy testing	Dialogue, debate, examples, guided discovery	
Techniques for avoiding, detecting and assessing vulnerabilities in memory corruption and specific to C language		
Techniques for avoiding, detecting and assessing vulnerabilities in the use and management of strings and meta-characters		

Techniques for avoiding, detecting and assessing vulnerabilities specific to the Linux operating system		
Penetration testing		
Penetration testing		
Penetration testing		

Bibliography
1. M. Down, J. McDonald, J. Schuh, „ The Art of Software Security Assessment. Identifying and Preventing Software Vulnerabilities ", Addison¬Wesley, 2007
2. M. Howard, D. LeBlanc, J. Viega, „ 24 Deadly Sins of Software Security. Programming Flows and How to Fix Them ", McGraw Hill, 2010
3. M. Howard, D. LeBlanc, „ Writing Secure Code for Windows Vista ", Microsoft Press, 2007
4. G. McGraw, „ Software Security:Building Security In ", Addison¬Wesley, 2006
5. R. Seacord, „CERT C Coding Standard: 98 Rules for Developing Safe, Reliable, and Secure Systems", Addison¬Wesley, 2 nd edition, 2014
6. , „ Common Weaknesses Enumeration (WCE)", on¬line: http://cwe.mitre.org/data/index.html

9. Corroborating the content of the discipline with the expectations of the epistemic community, professional associations and representative employers within the field of the program

It is carried out through regular discussions with representatives of significant employers in the field of information security.

Courses on security issues in application development and related fields (e.g. penetration tests) are present in many other masters in the field of computer and information security, at universities in the country and abroad, such as:

· Security of software systems, Master of Information Security, Al. I. Cuza, Iași, Faculty of Computers, http://profs.info.uaic.ro/~webdata/planuri/master/MISS1FS03.pdf

· Security of systems and applications, Master of Information Technology Security, Military Technical Academy, Bucharest, http://mta.ro/masterat/masterinfosec/curricula2013.html

· Secure Software Systems, Master of Science in Information Security, Carnegie Mellon University, USA, http://www.ini.cmu.edu/degrees/msis/courses.html

· Software Security, Master in Information Security, Royal Holloway University of London, Information Security Group,

https://www.royalholloway.ac.uk/isg/documents/pdf/coursespecs(msc)/modules201314/iy5607softwaresecurityspec1314.pdf

10. Evaluation

Activity type	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Percentage of final grade
10.4 Course	Ability to define concepts specific to security issues at source code level and to set out the methods for correctly evaluating and developing a source code from a security perspective. Ability to solve problems specific to the field.	written exam	60%
10.5 Seminar/laboratory	Ability to solve problems specific to the field Atendance, (inter) activity during laboratory / project hours.	practical work	20%(work dudring the semester)+20%(test seminar)
10.6 Minimum standard of performance			
• Ability to define fundamental software vulnerabilities, such as: buffer overflow, SQL code injection, XSS, etc.			

- Ability to identify fundamental software vulnerabilities and correct code (demonstrated in lab exercises and final evaluation).
- Grade: minimum 5 at each grading activity.
- Attendances: 75% attendance at seminar activities.
- Students with more than 2 unmotivated absences at the seminar will not be able to take the exam in the normal session and or in the retake examination session (seminar activities are activities that take place on the following principle "activity along the semester ", and they cannot be recovered or repeated for a possible retake examination (these students will have to repeat this course in the next academic year)). Students with medical certificates for each of their absences are exempted from this rule.

11. Labels ODD (Sustainable Development Goals)[2]

Not applicable.

Date: 13.04.2025	Signature of course coordinator Assoc.prof.phd. Mihai SUCIU	Signature of seminar coordinator Assoc.prof.phd. Mihai SUCIU
Date of approval:		Signature of the head of department Assoc.prof.phd. Adrian STERCA

[2] Keep only the labels that, according to the *Procedure for applying ODD labels in the academic process*, suit the discipline and delete the others, including the general one for *Sustainable Development* – if not applicable. If no label describes the discipline, delete them all and write „*Not applicable*.".